YouTube Video Link

YouTube Video Link

KG Coding

Some Other One shot Video Links:

- Complete HTML
- Complete CSS
- Complete JavaScript
- Complete React and Redux
- One shot University Exam Series

http://www.kgcoding.in/

Our YouTube Channels

KG Coding Android App

KG Coding

Knowledge GATE

KG Placement Prep

Sanchit Socket

1. Introduction to Java

1. Why you must learn Java
2. What is a Programming Language
3. What is an Algorithm
4. What is Syntax
5. History of Java
6. Magic of Byte Code
7. How Java Changed the Internet
8. Java Buzzwords
9. Object Oriented Programming

1.1 Why you must learn Java

1. One of the most popular language. Java currently runs on 60,00,00,00,000 devices.
2. Wide Usage (Web-apps, backend, Mobile apps, enterprise software).
3. High paying and a lot of Jobs
4. Object Oriented
5. Rich APIs and Community Support

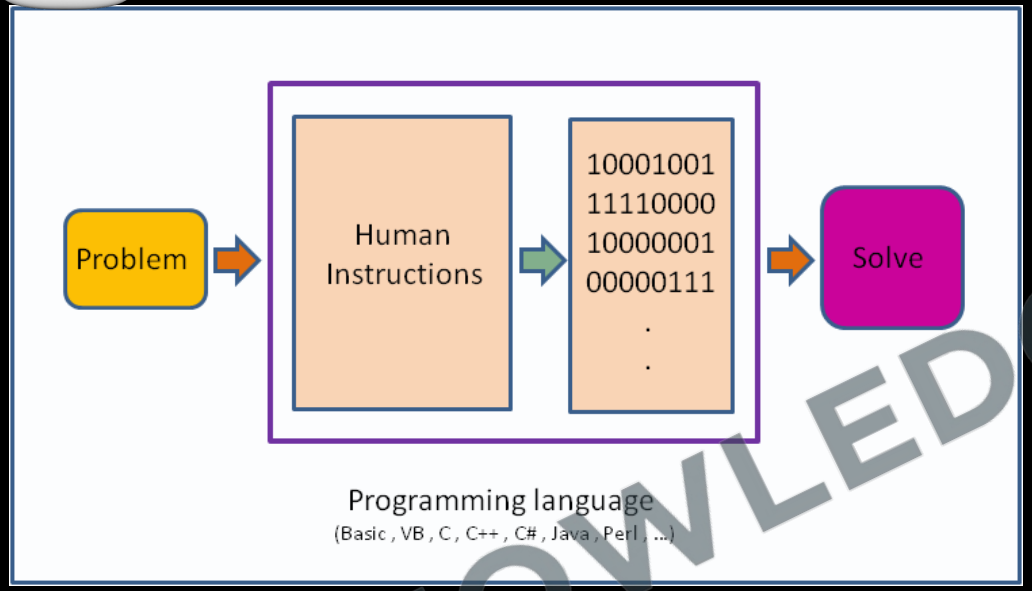
1.2 What is a Programming Language

Humans use natural language (like Hindi/English) to communicate

1.2 What is a Programming Language

Computers only understand 0/1 or on/off

1.2 What is a Programming Language



- Giving instructions to a computer
- **Instructions**: Tells computer what to do.
- These instructions are called code.
- Human instructions are given in High level languages.

Compiler converts high level languages to low level languages or machine code.

1.3 What is an Algorithm

1.3 What is an Algorithm

An algorithm is a step-by-step procedure for solving a problem or performing a task.

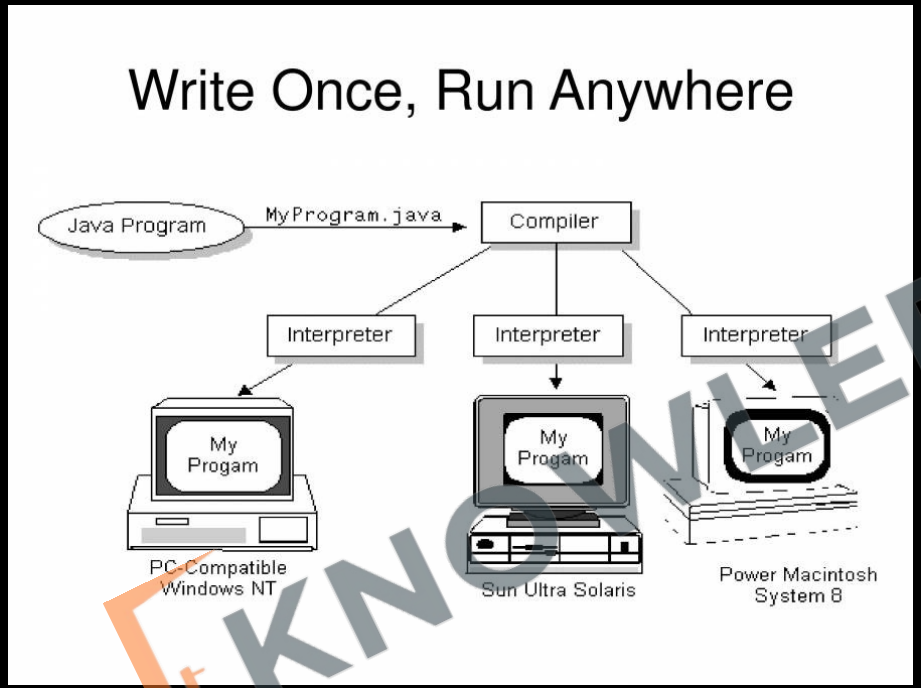
1.4 What is Syntax

- Structure of words in a sentence.
- Rules of the language.
- For programming exact syntax must be followed.

1.5 History of Java

Developed by James Gosling at Sun Microsystems (Early 1990s):
Originally named 'Oak', later renamed Java in 1995.

1.5 History of Java

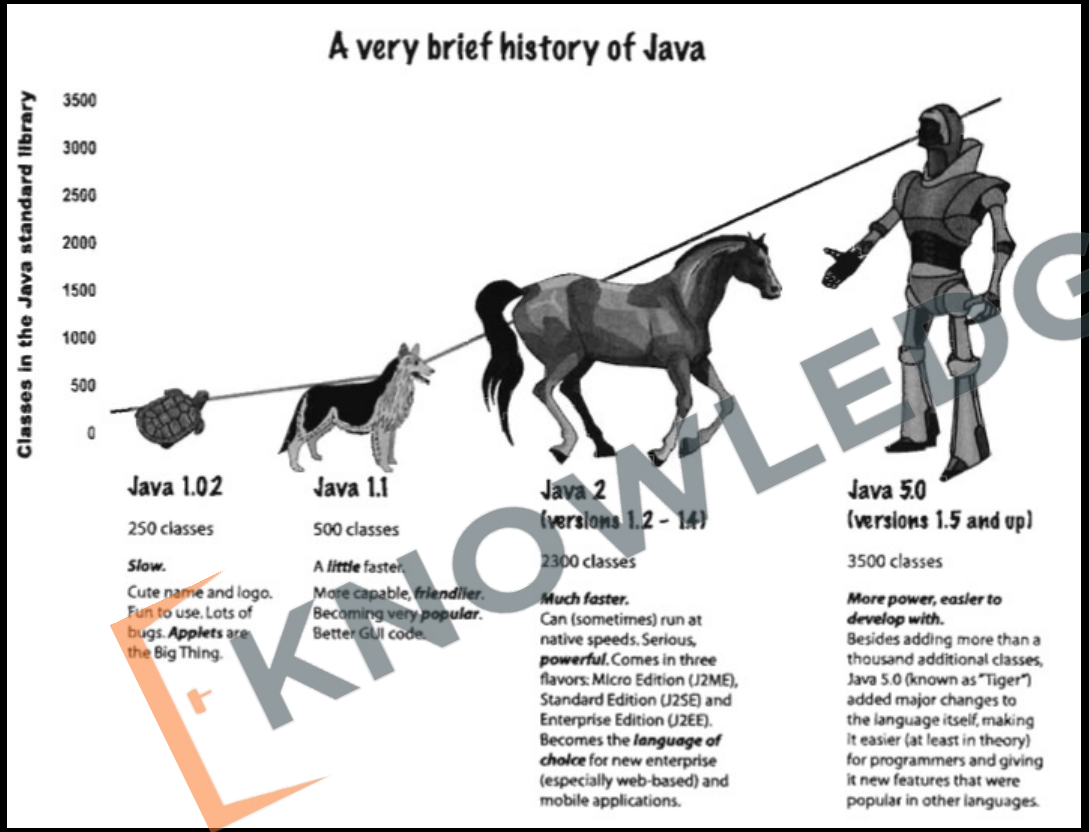


First Release (1995): Introduced "Write Once, Run Anywhere" concept with cross-platform compatibility.

1.5 History of Java

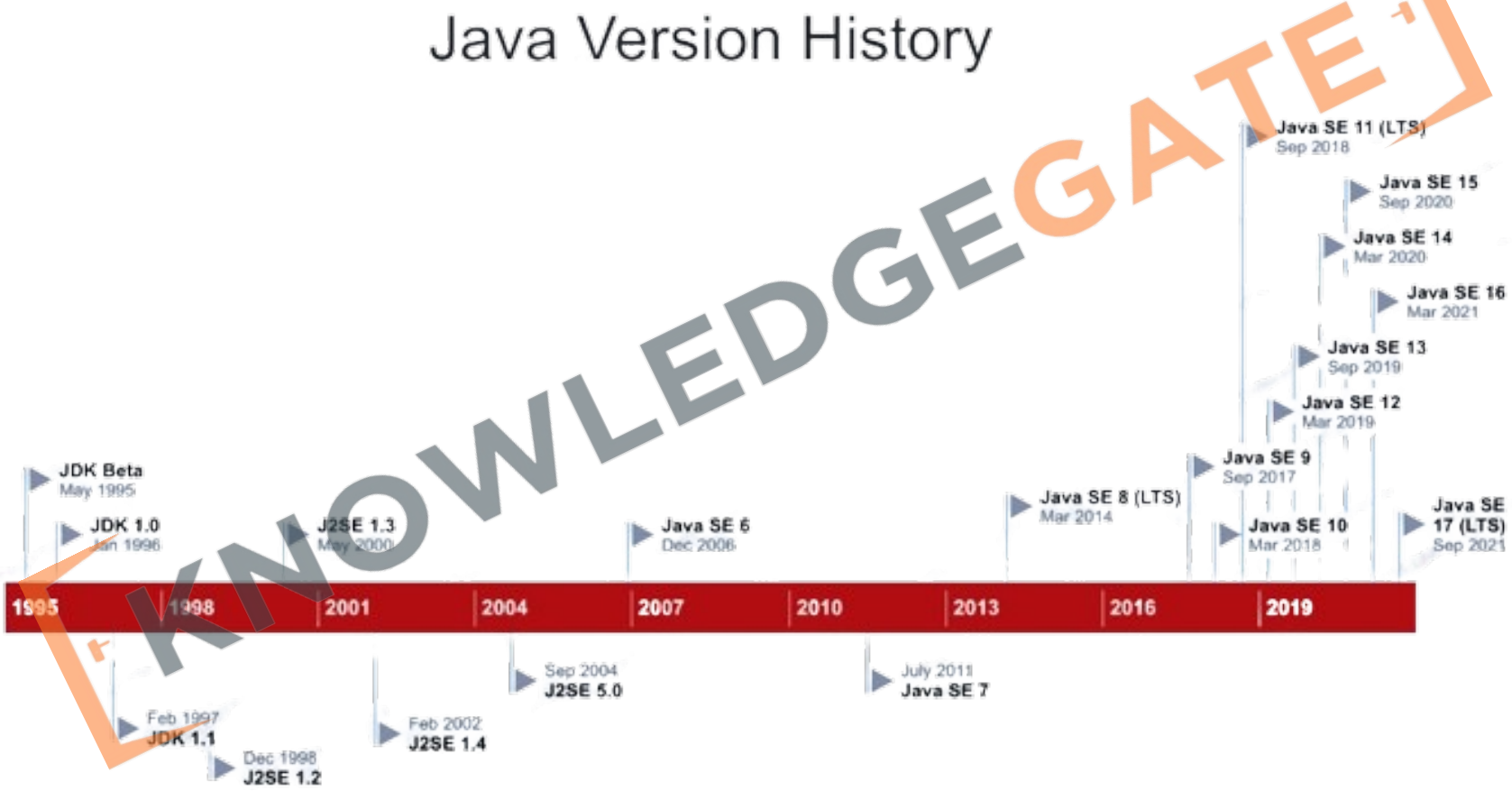
Developed with vision of backward compatibility. Should not break with new version release.

1.5 History of Java

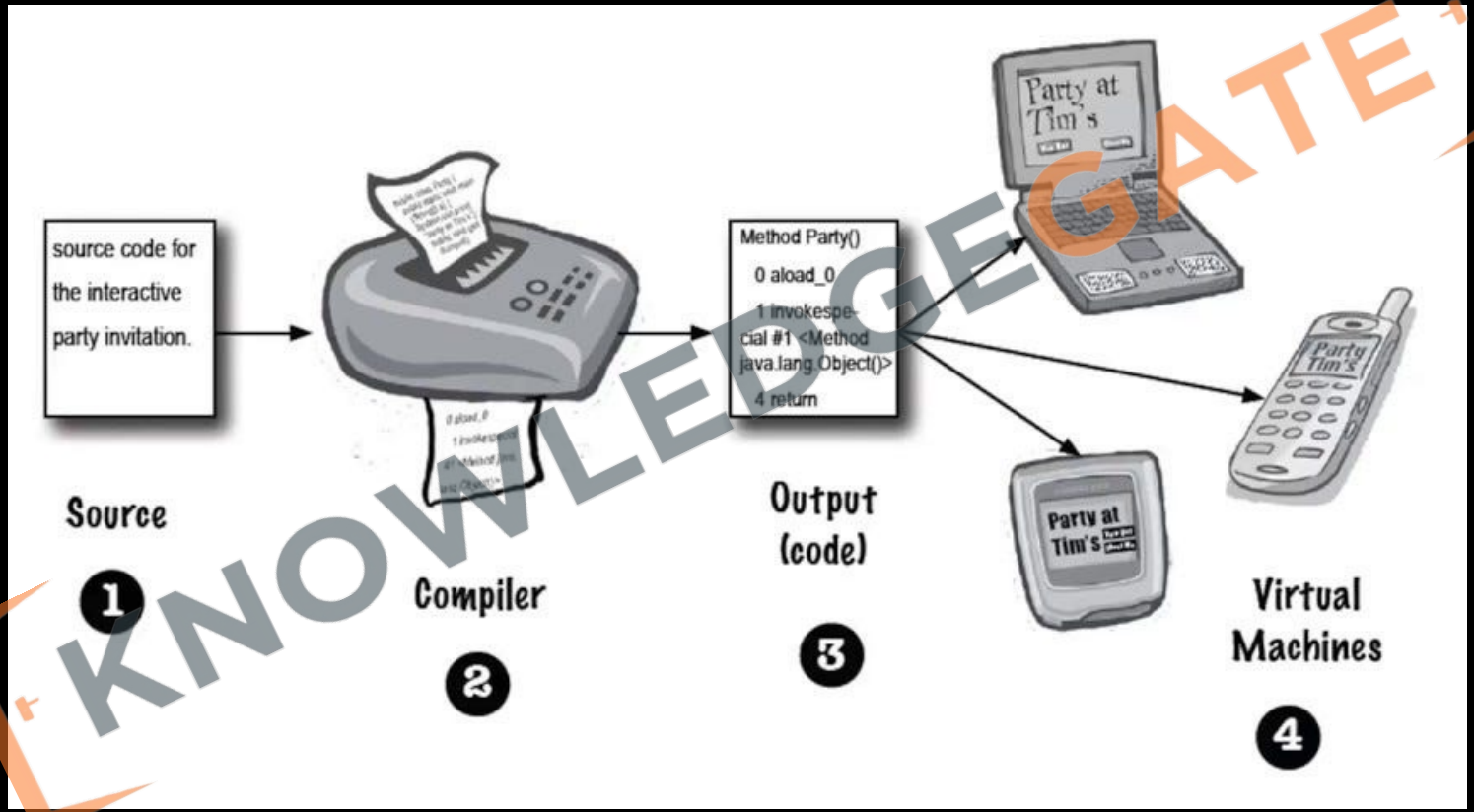


Rapid Growth and Diversification (Late 1990s - 2010s): Expanded from web applets to server-side applications; standardized into different editions for various computing platforms.

1.5 History of Java



1.6 Magic of Byte Code



1.7 How Java Changed the Internet

Portability with Write Once Run Anywhere

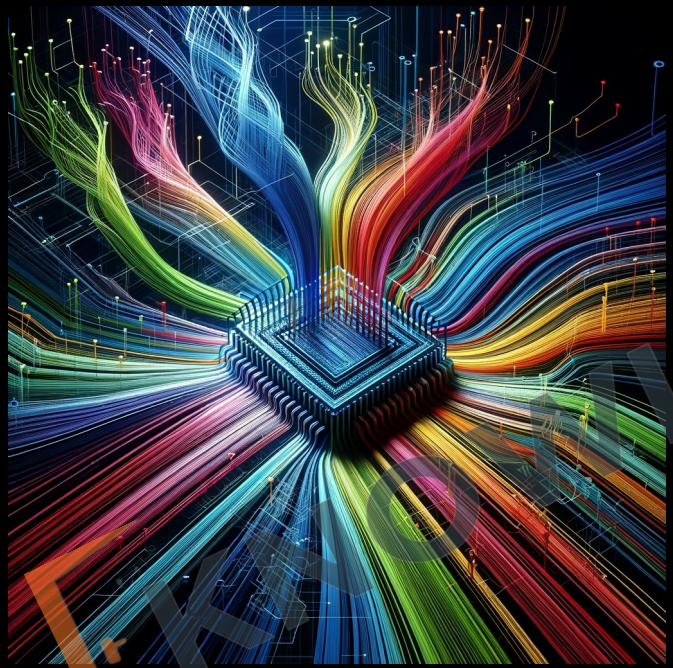
Security because of Code running on Virtual Machine

1.8 Java Buzzwords

Robust

Java is robust due to its strong memory management, exception handling, and type-checking mechanisms, which help in preventing system crashes and ensuring reliable performance.

1.8 Java Buzzwords



Multithreaded

Multithreading in programming is the ability of a CPU to execute multiple threads concurrently, allowing for more efficient processing and task management.

1.8 Java Buzzwords

Architecture Neutral

Java is architecturally neutral because its compiled code (bytecode) can run on any device with a Java Virtual Machine (JVM), regardless of the underlying hardware architecture.

1.8 Java Buzzwords

Interpreted and High Performance

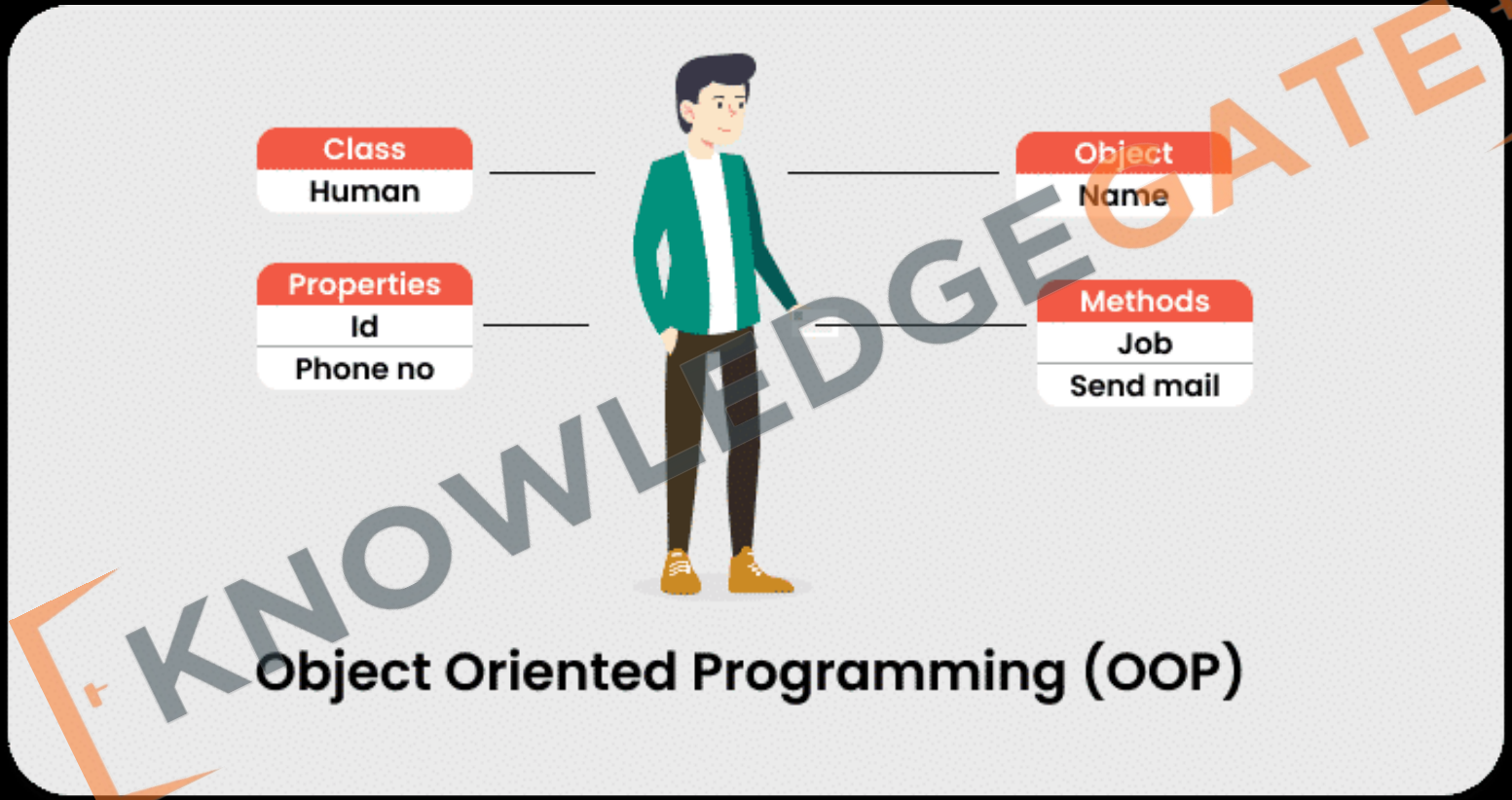
Java combines high performance with interpretability, as its bytecode is interpreted by the Java Virtual Machine (JVM), which employs Just-In-Time (JIT) compilation for efficient and fast execution.

1.8 Java Buzzwords

Distributed

Java is inherently distributed, designed to facilitate network-based application development and interaction, seamlessly integrating with Internet protocols and remote method invocation.

1.9 Object Oriented Programming



1.9 Object Oriented Programming

Revision

1. Why you must learn Java
2. What is a Programming Language
3. What is an Algorithm
4. What is Syntax
5. History of Java
6. Magic of Byte Code
7. How Java Changed the Internet
8. Java Buzzwords
9. Object Oriented Programming

KG Coding

Some Other One shot Video Links:

- Complete HTML
- Complete CSS
- Complete JavaScript
- Complete React and Redux
- One shot University Exam Series

http://www.kgcoding.in/

Our YouTube Channels

KG Coding Android App

KG Coding

Knowledge GATE

KG Placement Prep

Sanchit Socket

2. Java Basics

1. Installing JDK
2. First Class using Text Editor
3. Compiling and Running
4. Anatomy of a Class
5. File Extensions
6. JDK vs JVM vs JRE
7. Showing Output
8. Importance of the main method
9. Installing IDE(Intellij Idea)
10. Creating first Project

2.1 Installing JDK

- Search JDK Download
- Make sure to download from the oracle website.
- Download the latest version

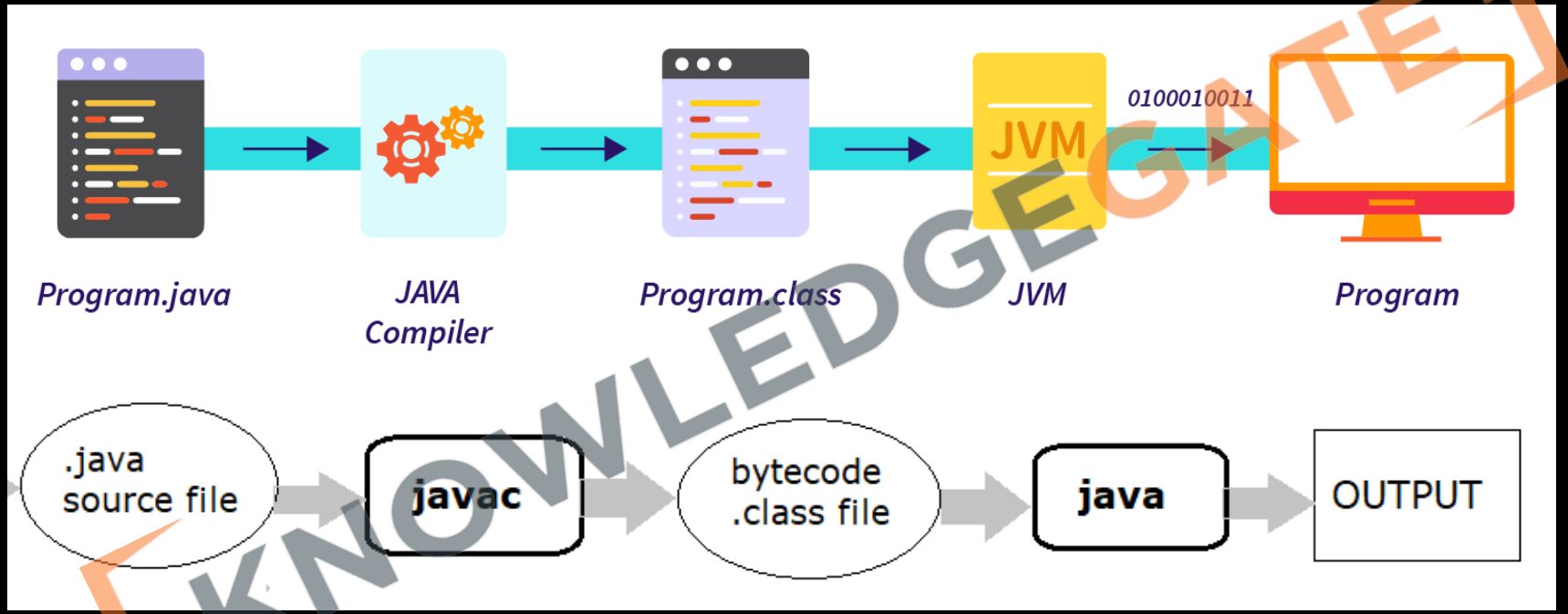
2.2 First Class using Text Editor

First.java

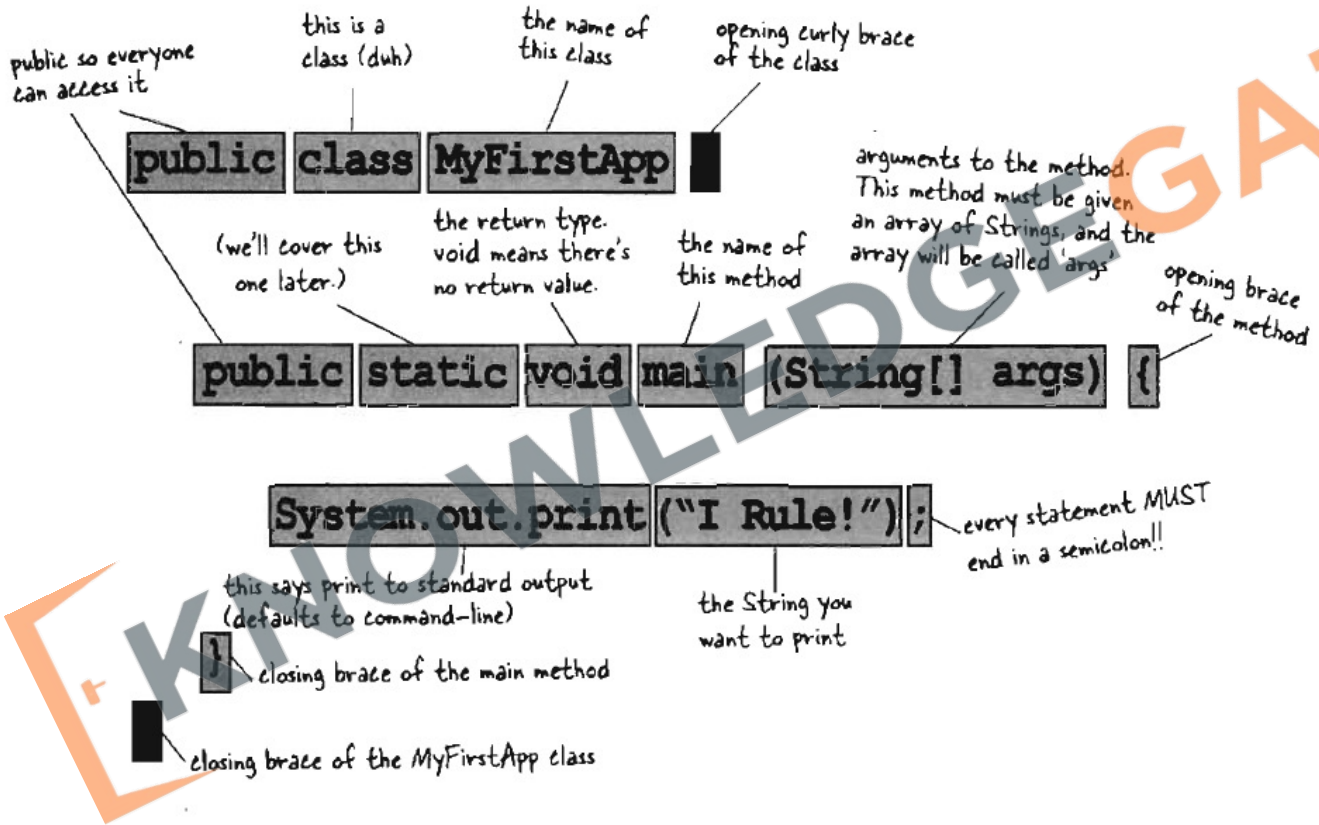
```
import java.lang.*;

public class First {
  public static void main(String[] args) {
    System.out.println("Welcome to KGCoding");
  }
}
```

2.3 Compiling and Running



2.4 Anatomy of a Class



2.5 File Extensions

.Java

- Contains Java Source Code
- High Level Human Readable
- Used for Development
- File is editable

.Class

- Contains Java Bytecode
- For consumption of JVM
- Used for Execution
- Not meant to be edited

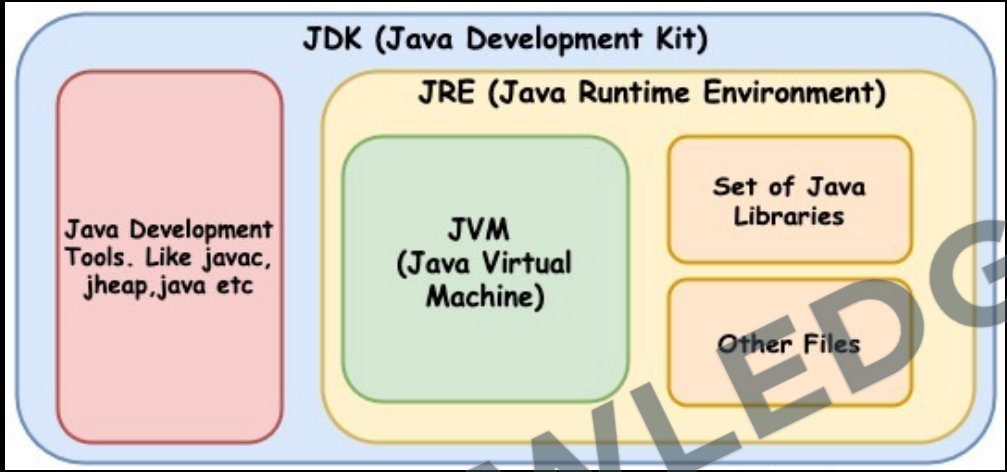
Main.java

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

Main.class

```
.∫æ
java/lang/Object<init>()V
java/lang/SystemoutLjava/io/PrintStream;
Hello world!
java/io/PrintStreamprintln(Ljava/lang/
String;)VMainCodeLineNumberTableLocalVariableTablethisLMain;ma
in([Ljava/lang/String;)Vargs[Ljava/lang/String;
SourceFile      Main.java!/*∑±
        7       ≤
∂
        !
```

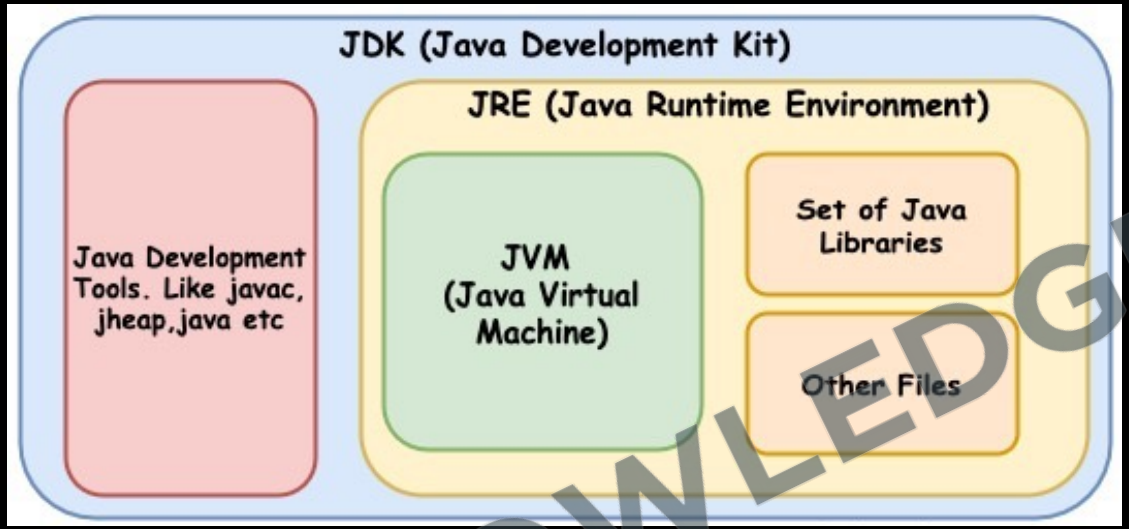
2.6 JDK vs JVM vs JRE



.JDK

- It's a software development kit required to develop Java applications.
- Includes the JRE, an interpreter/loader (Java), a compiler (javac), a doc generator (Javadoc), and other tools needed for Java development.
- Essentially, JDK is a superset of JRE.

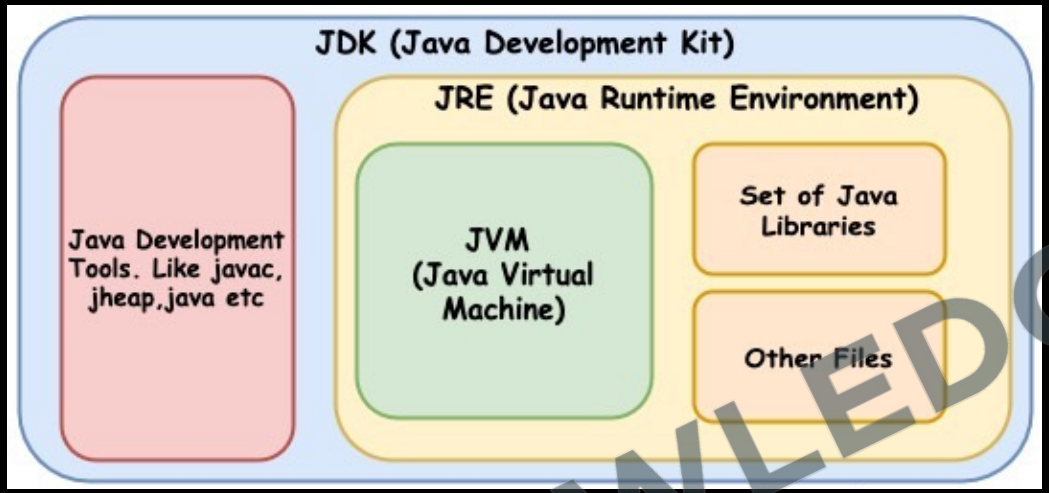
2.6 JDK vs JVM vs JRE



.JRE

- It's a part of the JDK but can be downloaded separately.
- Provides the libraries, the JVM, and other components to run applications
- Does not have tools and utilities for developers like compilers or debuggers.

2.6 JDK vs JVM vs JRE



.JVM

- It's a part of JRE and responsible for executing the bytecode.
- Ensures Java's write-once-run-anywhere capability.
- Not platform-independent: a different JVM is needed for each type of OS.

2.7 Showing Output

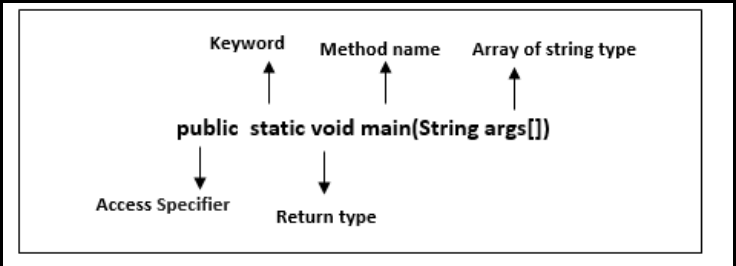
Example	Result

`System.out.print("one");`
`System.out.print("two");`
`System.out.println("three");`
	onetwothree

`System.out.println("one");`
`System.out.println("two");`
`System.out.println("three");`
	one
	two
	three

`System.out.println();`	[new line]

2.8 Importance of the main method



- Entry Point: It's the entry point of a Java program, where the execution starts. Without the main method, the Java Virtual Machine (JVM) does not know where to begin running the code.
- Public and Static: The main method must be public and static, ensuring it's accessible to the JVM without needing to instantiate the class.
- Fixed Signature: The main method has a fixed signature: public static void main(String[] args). Deviating from this signature means the JVM won't recognize it as the starting point.

KG Coding

Some Other One shot Video Links:

- Complete HTML
- Complete CSS
- Complete JavaScript
- Complete React and Redux
- One shot University Exam Series

http://www.kgcoding.in/

Our YouTube Channels

KG Coding Android App

KG Coding

Knowledge GATE

KG Placement Prep

Sanchit Socket

2.9 What is IDE

1. IDE stands for Integrated Development Environment.
2. Software suite that consolidates basic tools required for software development.
3. Central hub for coding, finding problems, and testing.
4. Designed to improve developer efficiency.

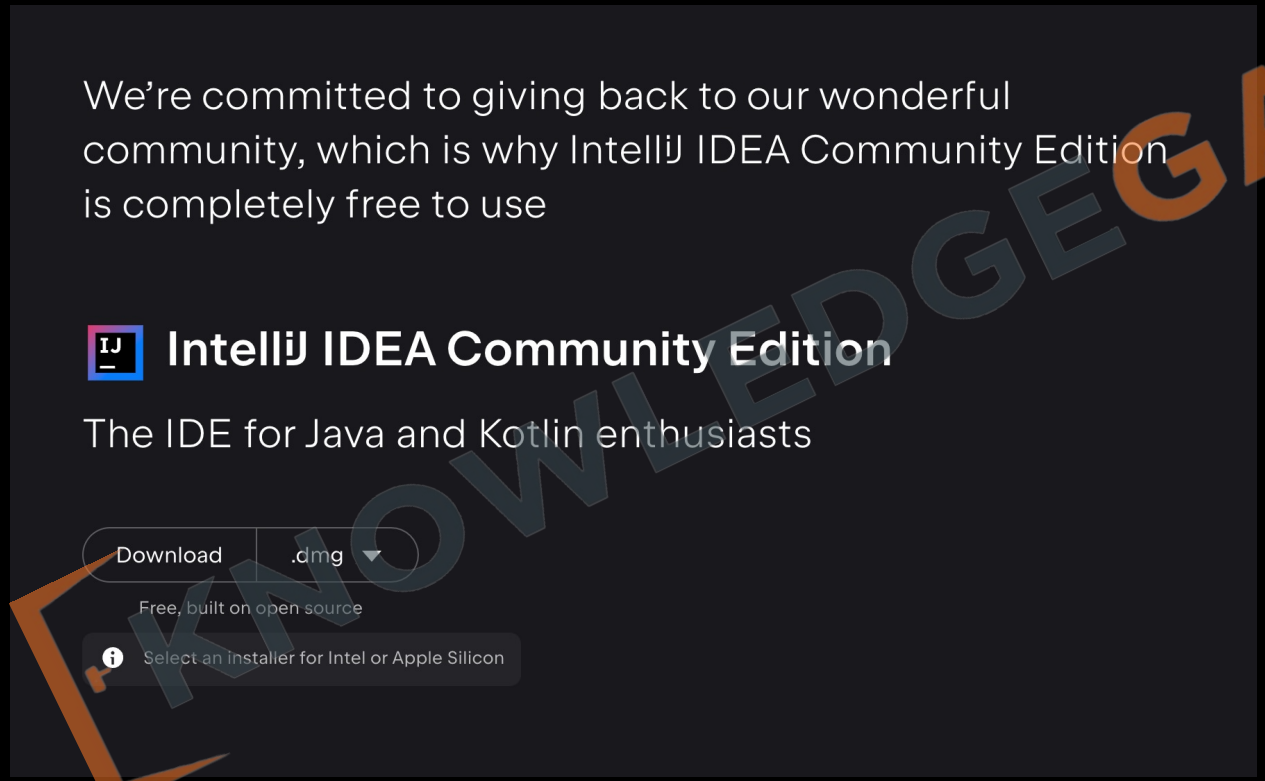
2.9 Need of IDE

1. Streamlines development.
2. Increases productivity.
3. Simplifies complex tasks.
4. Offers a unified workspace.
5. IDE Features
    1. Code Autocomplete
    2. Syntax Highlighting
    3. Version Control
    4. Error Checking

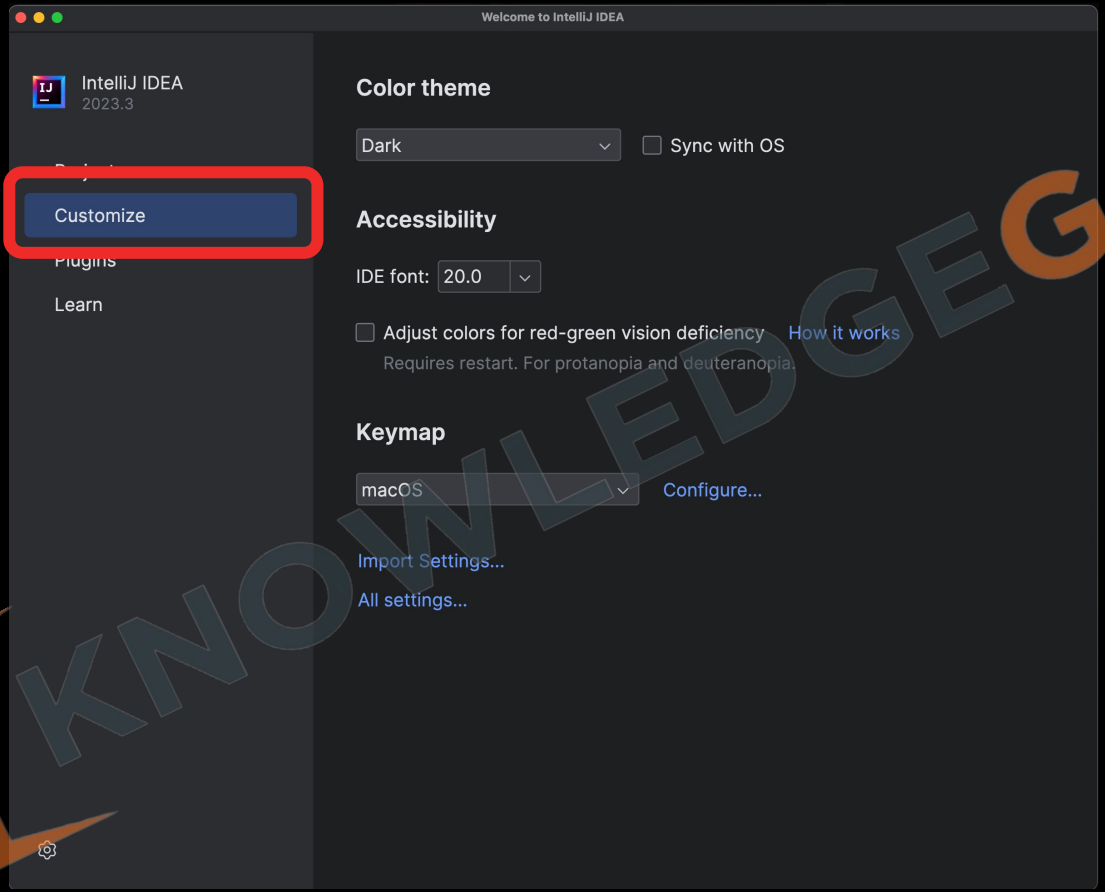
2.9 Installing IDE(Intellij Idea)

- Search Intellij IDEA
- Make sure to download the community Version.
- Keep Your Software up to date.

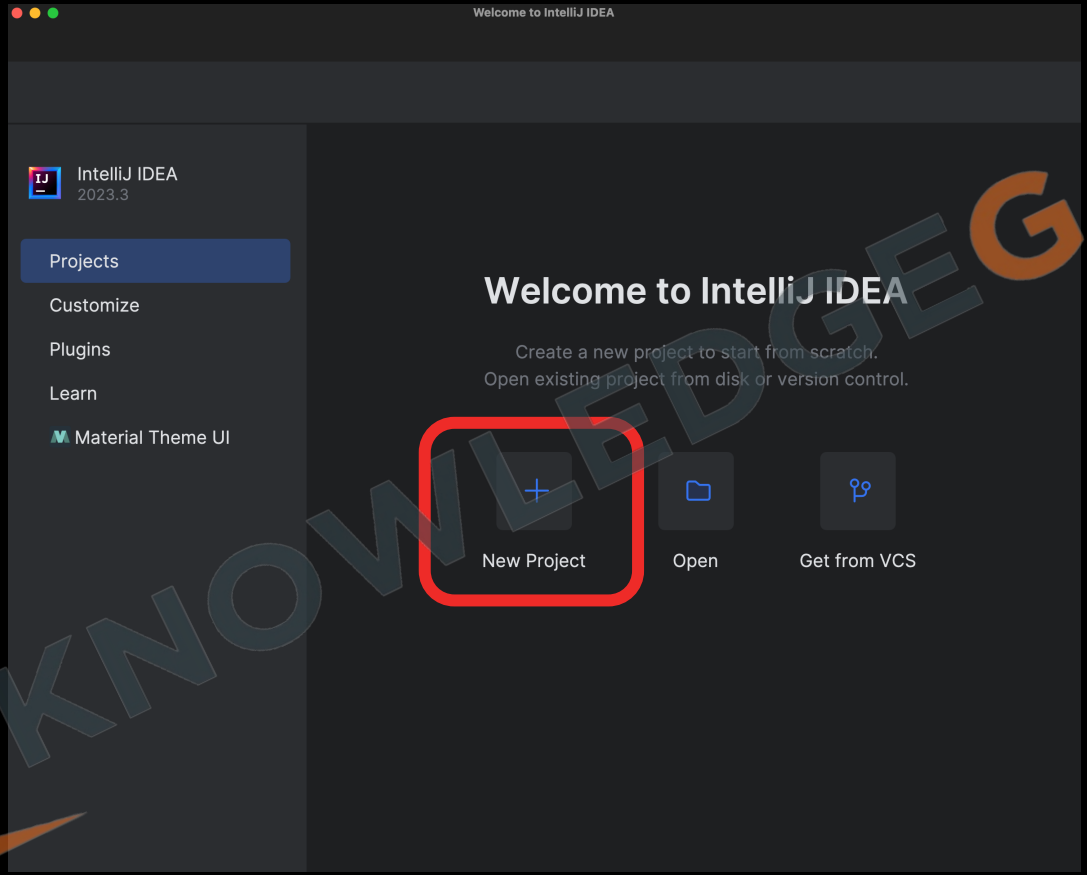
2.9 Installing IDE(Intellij Idea)



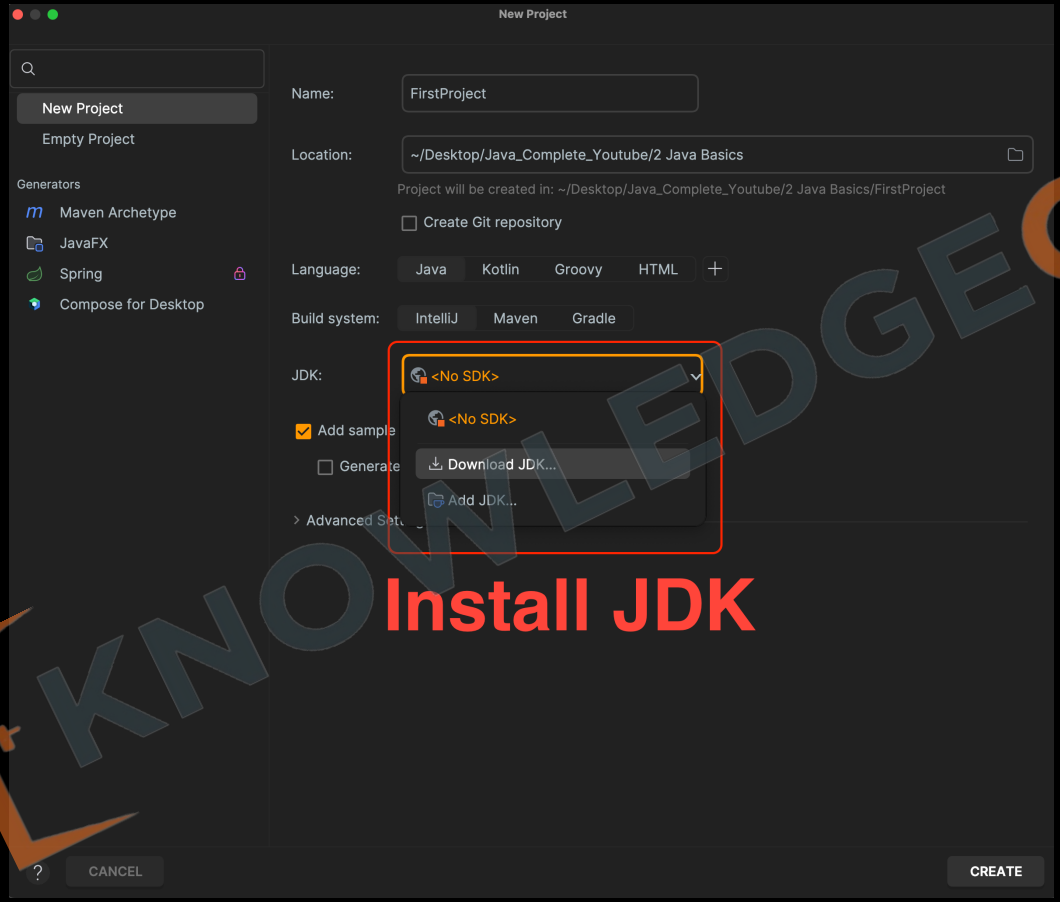
2.9 Installing IDE(Intellij Idea)



2.9 Installing IDE(Intellij Idea)

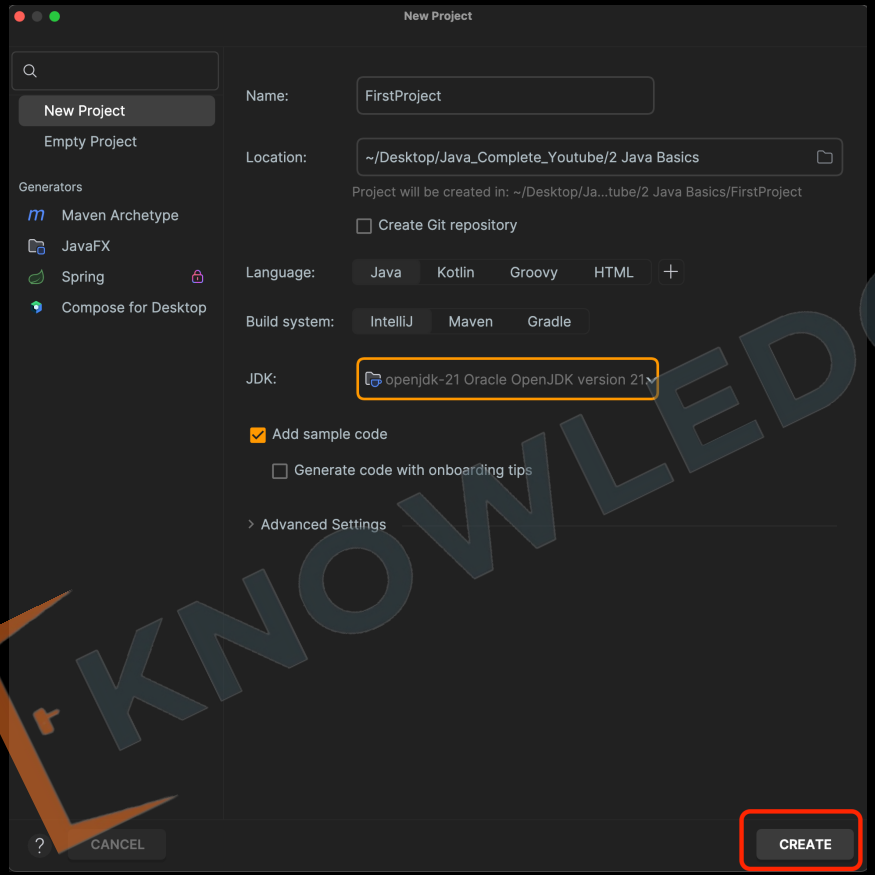


2.9 Installing IDE(Intellij Idea)



Install JDK

2.10 Creating first Project

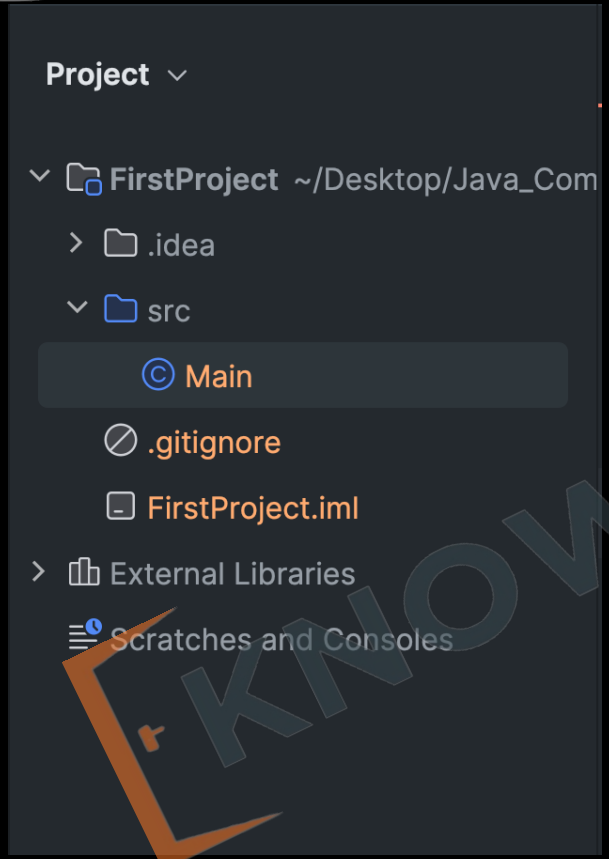


2.10 First Java Class

Main.java

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

2.10 Project Structure

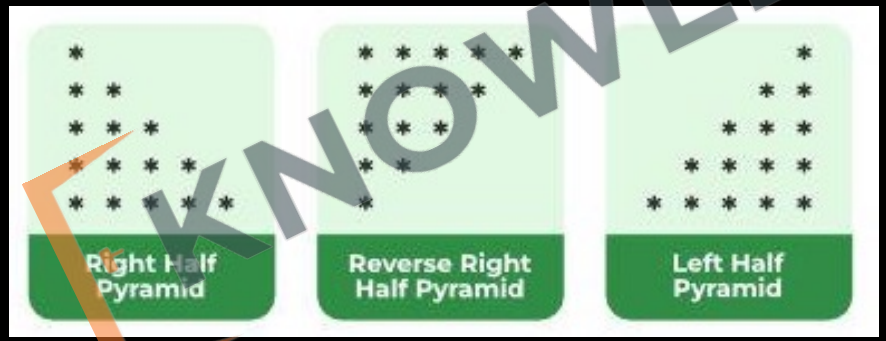


Revision

1. Installing JDK
2. First Class using Text Editor
3. Compiling and Running
4. Anatomy of a Class
5. File Extensions
6. JDK vs JVM vs JRE
7. Showing Output
8. Importance of the main method
9. Installing IDE(Intellij Idea)
10. Creating first Project

Tasks:

1. Create a class to output “good morning” using a text editor and check output.
2. Create a new Project in Intelij Idea and output “subscribe” on the console.
3. Show the following patterns:



Practice Exercise

Java Basics

Answer in True/False:

1. Computers understand high level languages like Java, C.
2. An Algorithm is a set of instructions to accomplish a task.
3. Computer is smart enough to ignore incorrect syntax.
4. Java was first released in 1992.
5. Java was named over a person who made good coffee.
6. ByteCode is platform independent.
7. JDK is a part of JRE.
8. It's optional to declare main method as public.
9. .class file contains machine code.
10. println adds a new line at the end of the line.

Practice Exercise

Java Basics

Answer in True/False:

1. Computers understand high level languages like Java, C. — False
2. An Algorithm is a set of instructions to accomplish a task. — True
3. Computer is smart enough to ignore incorrect syntax. — False
4. Java was first released in 1992. — False
5. Java was named over a person who made good coffee. — False
6. ByteCode is platform independent. — True
7. JDK is a part of JRE. — False
8. It's optional to declare main method as public. — False
9. .class file contains machine code. — False
10. println adds a new line at the end of the line. — True

KG Coding

Some Other One shot Video Links:

- Complete HTML
- Complete CSS
- Complete JavaScript
- Complete React and Redux
- One shot University Exam Series

http://www.kgcoding.in/

Our YouTube Channels

KG Coding Android App

KG Coding

Knowledge GATE

KG Placement Prep

Sanchit Socket

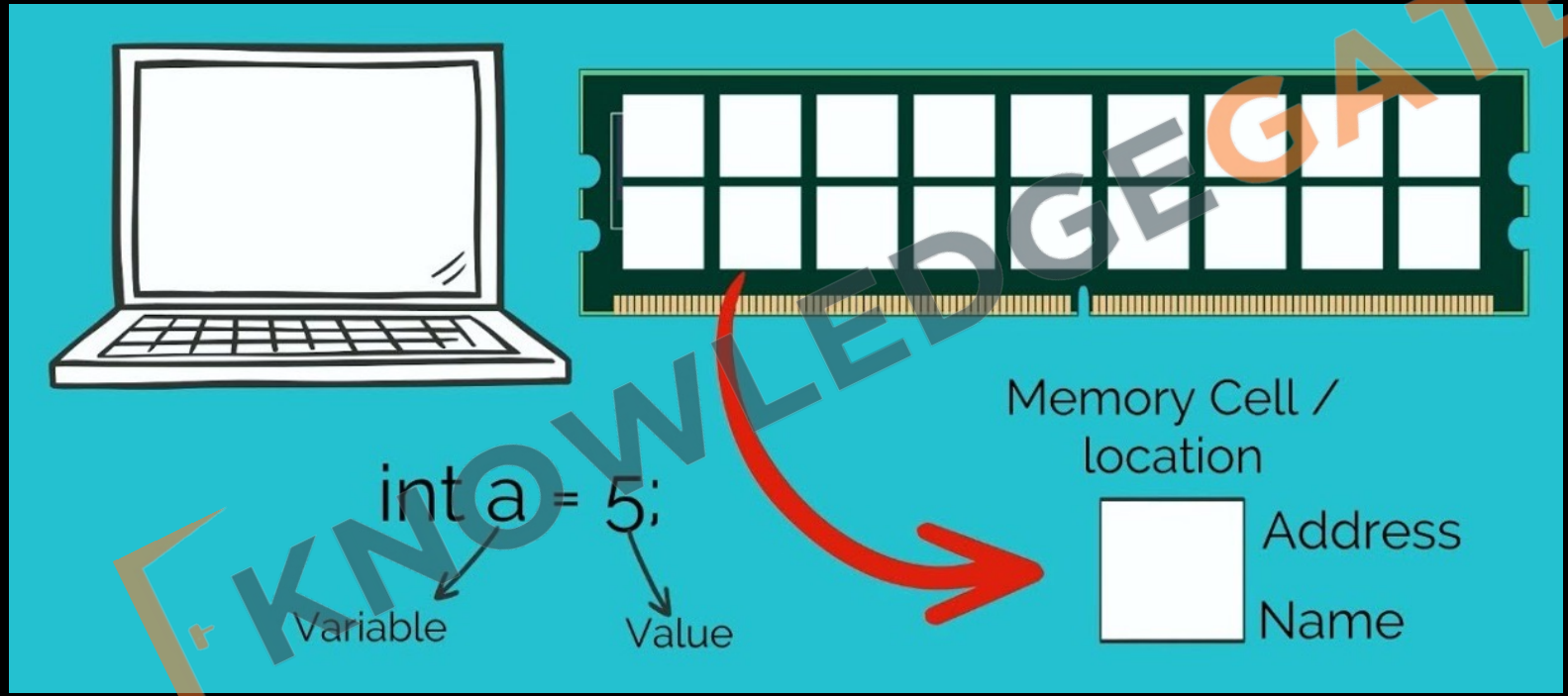
3 Data Types, Variables & Input

1. Variables
2. Data Types
3. Naming Conventions
4. Literals
5. Keywords
6. Escape Sequences
7. User Input
8. Type Conversion and Casting

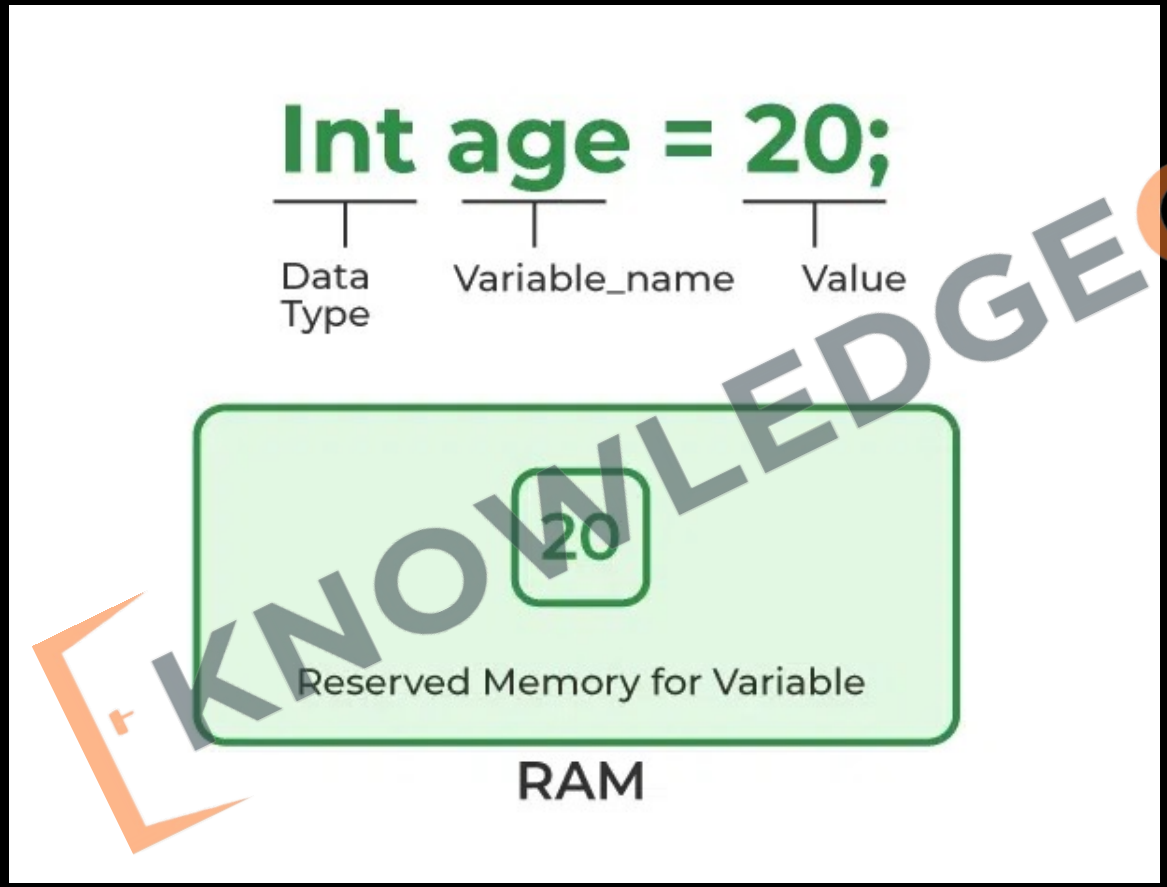
3.1. What are Variables?

Variables are like containers used for storing data values.

3.1. Memory Allocation



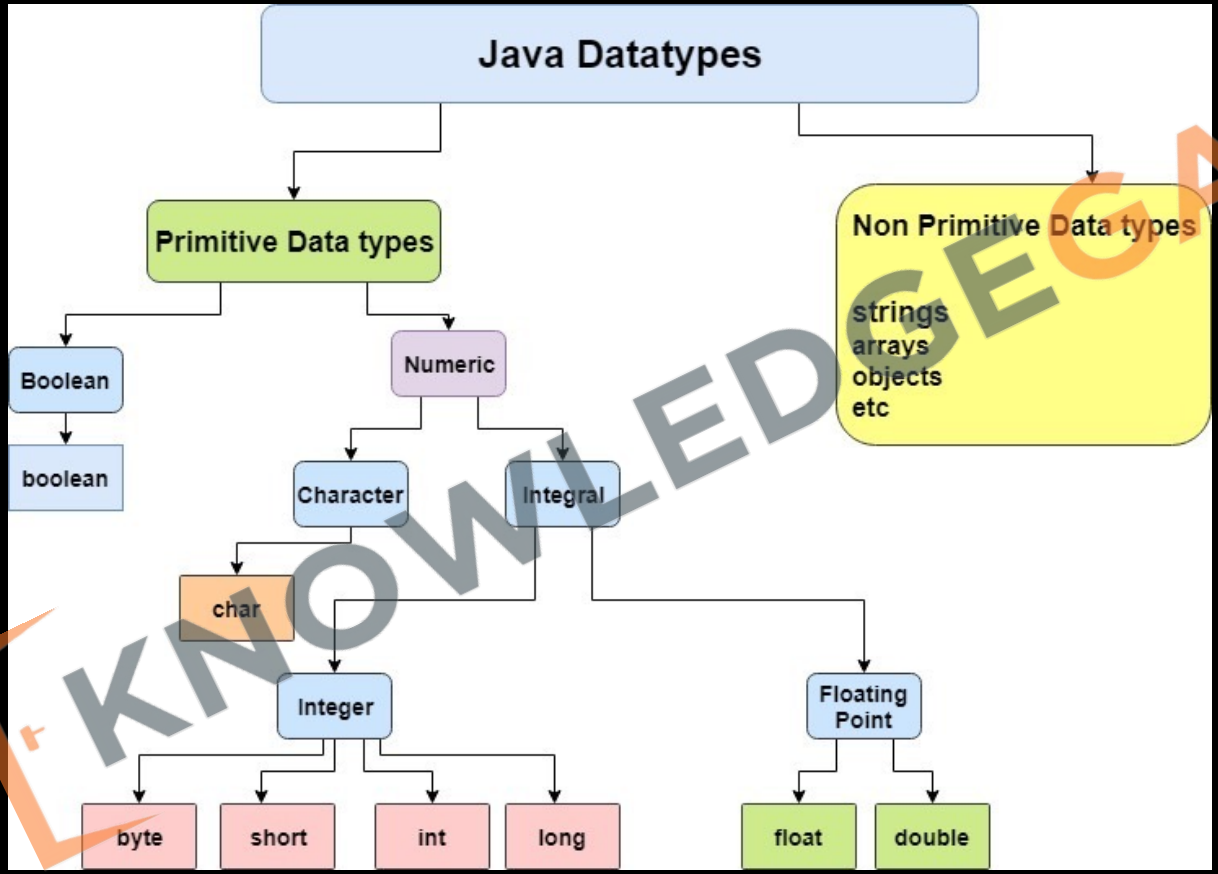
3.1. Variable Declaration



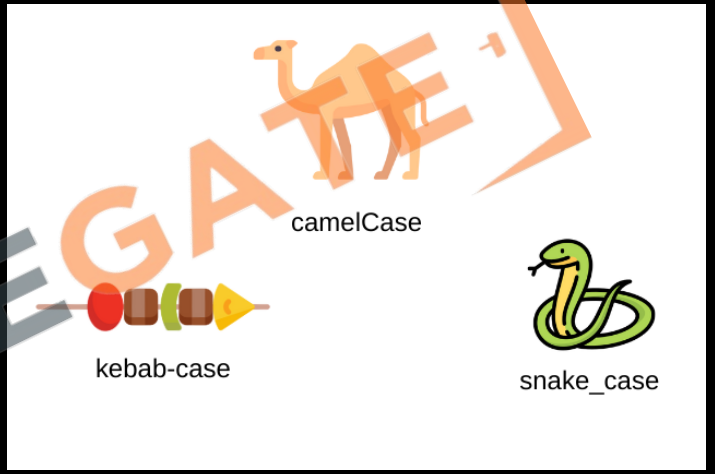
3.2 Data Types

	Size	Default Value	Type of Value Stored
byte	1 byte	0	Integral
short	2 byte	0	Integral
int	4 byte	0	Integral
long	8 byte	0L	Integral
char	2 byte	'\u0000'	Character
float	4 byte	0.0f	Decimal
double	8 byte	0.0d	Decimal
boolean	1 bit (till JDK 1.3 it uses 1 byte)	false	True or False

3.2 Data Types



3.3. Naming Conventions



camelCase

- Start with a lowercase letter. Capitalize the first letter of each subsequent word.
- Example: myVariableName

snake_case

- Start with an lowercase letter. Separate words with underscore
- Example: my_variable_name

Kebab-case

- All lowercase letters. Separate words with hyphens. Example: my-variable-name

Keep a Good and Short Name

- Choose names that are descriptive but not too long. It should make it easy to understand the variable's purpose.
- Example: age, firstName, isMarried

3.3. Java Identifier Rules

1. The only allowed characters for identifiers are all alphanumeric characters([A-Z],[a-z],[0-9]), '$' (dollar sign) and '_' (underscore).
2. Can't use keywords or reserved words
3. Identifiers should not start with digits([0-9]).
4. Java identifiers are case-sensitive.
5. There is no limit on the length of the identifier but it is advisable to use an optimum length of 4 – 15 letters only.

1)_$_ (valid)
2)Ca$h (valid)
3)Java2share (valid)
4)all@hands (invalid)
5)123abc (invalid)
6)Total# (invalid)
7)Int (valid)
8)Integer (valid)
9)int (invalid)
10) tot123

3.4 Literals

Integer literals -> 10, 5, -8 etc...

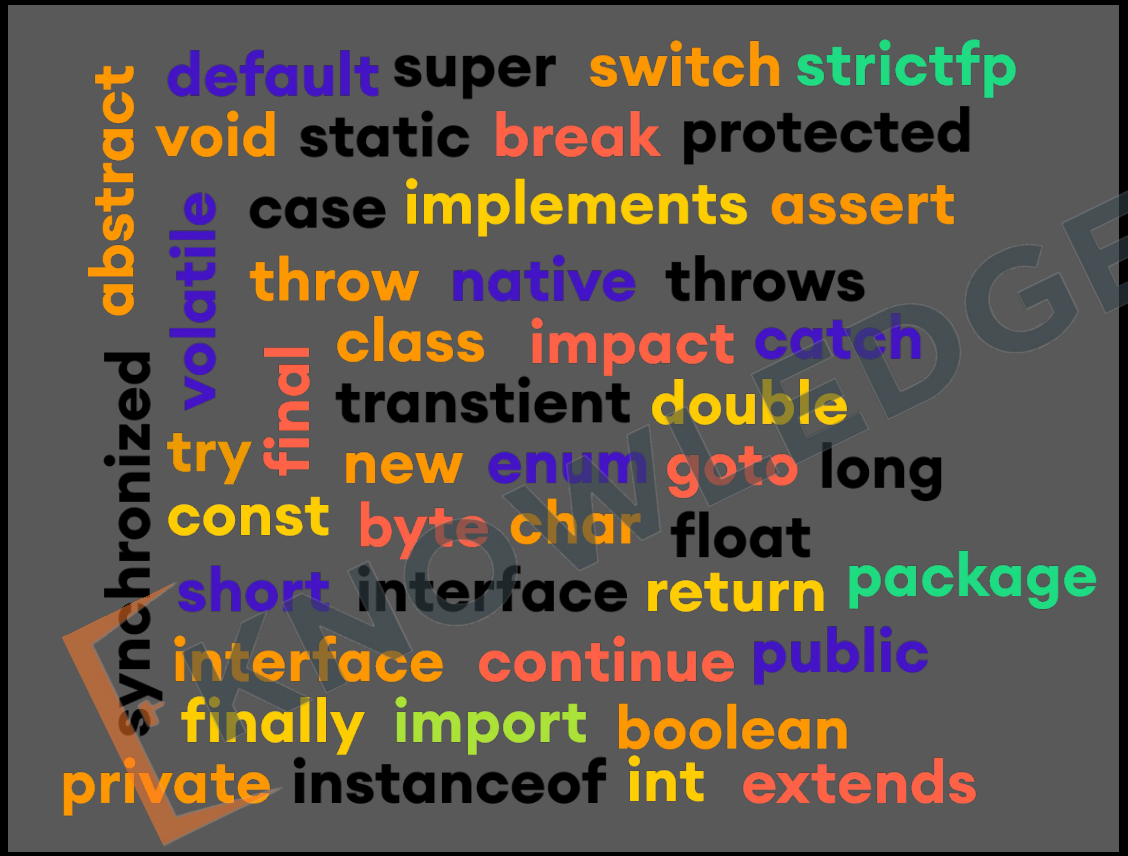
Floating-point literals -> 1.2, 0.25, -1.999, etc...

Boolean literals -> true, false

Character literals -> ‘a’, ‘A’, ‘N’, ‘q’, etc...

String literals -> “hi”, “hello”, “What’s up?”

3.5 Keywords



3.6 Escape Sequences

Escape Sequence	Description
\t	Insert a tab in the text at this point.
\b	Insert a backspace in the text at this point.
\n	Insert a newline in the text at this point.
\'	Insert a single quote character in the text at this point.
\"	Insert a double quote character in the text at this point.
\\	Insert a backslash character in the text at this point.

CHALLENGE

Task:

4. Show the following patterns using single print statement:

3.7 User Input

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your name:");
        String name = scanner.nextLine();
        System.out.println("Welcome " + name);
    }
}
```

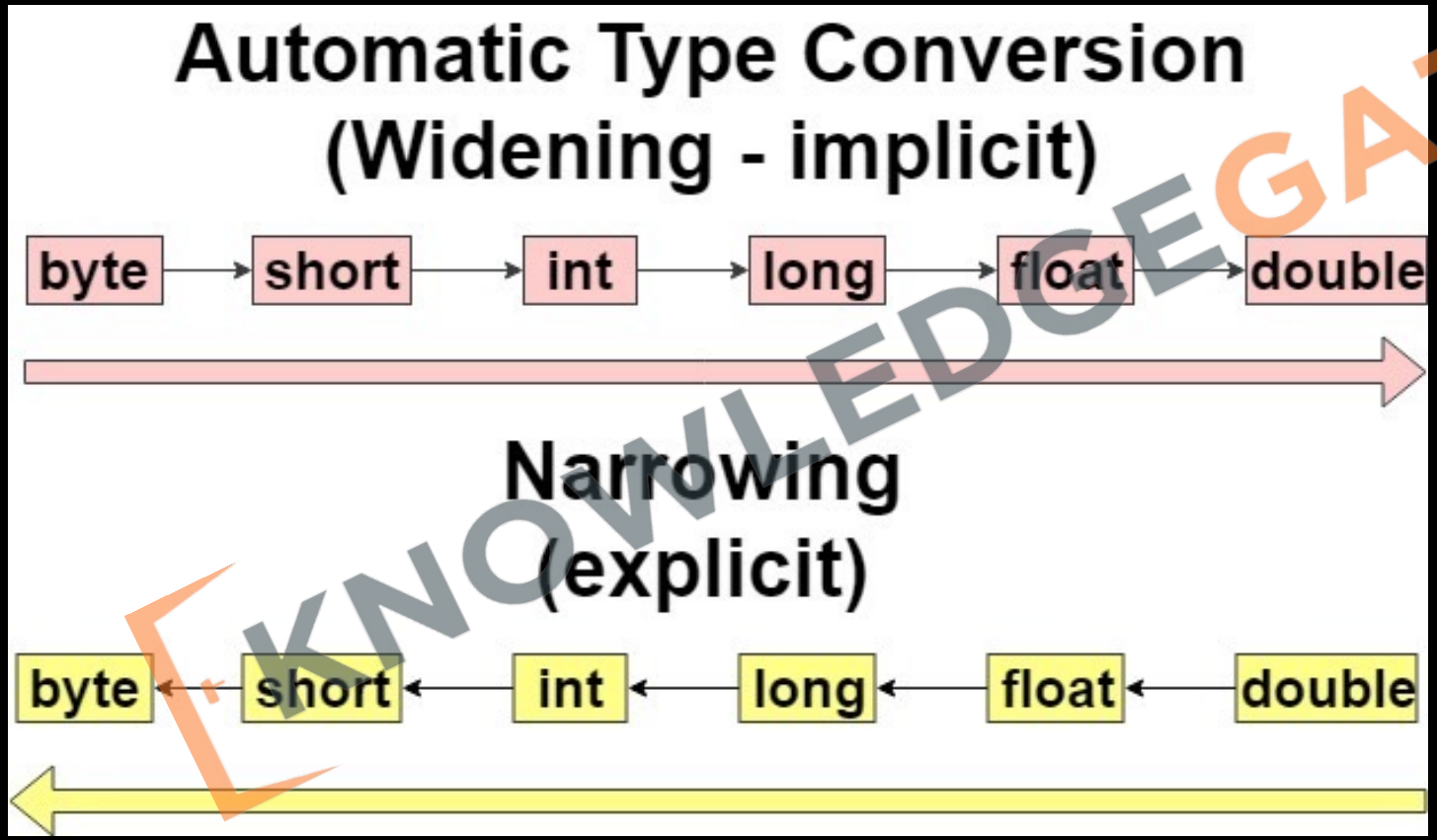
NAME
nextInt();
nextDouble();
nextFloat();
nextLong();
nextShort();
next();
nextLine();

CHALLENGE

5.Create a program to input name of the person and respond with ”Welcome NAME to KG Coding”

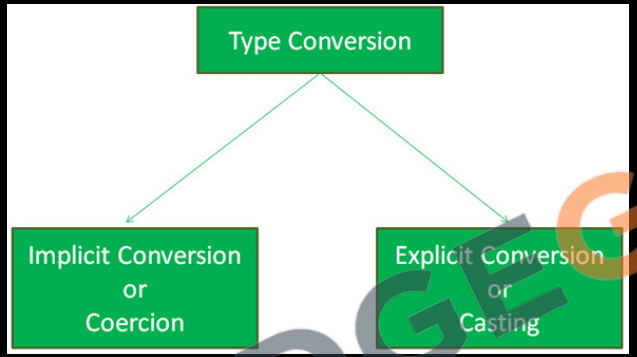
6.Create a program to add two numbers.

3.8 Type Conversion and Casting



3.8 Type Conversion and Casting

```java
// implicit
long big = 45;
float dec = 3;
double d = 3.4f;
```




```java
// explicit
float eDec = (float) 3.4;
long eBig = (long) 3.4;
int eInt = (int) 3.4;
```

Revision

1. Variables
2. Data Types
3. Naming Conventions
4. Literals
5. Keywords
6. Escape Sequences
7. User Input
8. Type Conversion and Casting

Practice Exercise

Java Basics

Answer in True/False:

1. In Java, a variable's name can start with a number.
2. char in Java can store a single character.
3. Class names in Java typically start with a lowercase letter.
4. 100L is a valid long literal in Java.
5. \d is an escape sequence in Java for a digit character.
6. Scanner class is used for reading console input.
7. In Java, an int can be automatically converted to a byte.
8. Java variable names are case-sensitive.
9. Scanner class can be used to read both primitive data types and strings.
10. Explicit casting is required to convert a double to an int.

Practice Exercise

Java Basics

Answer in True/False:

1. In Java, a variable's name can start with a number. — False
2. char in Java can store a single character. — True
3. Class names in Java typically start with a lowercase letter. — False
4. 100L is a valid long literal in Java. — True
5. \d is an escape sequence in Java for a digit character. — False
6. Scanner class is used for reading console input. — True
7. In Java, an int can be automatically converted to a byte. — False
8. Java variable names are case-sensitive. — True
9. Scanner class can be used to read both primitive data types and strings. — True
10. Explicit casting is required to convert a double to an int. — True

KG Coding

Some Other One shot Video Links:

- Complete HTML
- Complete CSS
- Complete JavaScript
- Complete React and Redux
- One shot University Exam Series

http://www.kgcoding.in/

Our YouTube Channels

KG Coding Android App

KG Coding

Knowledge GATE

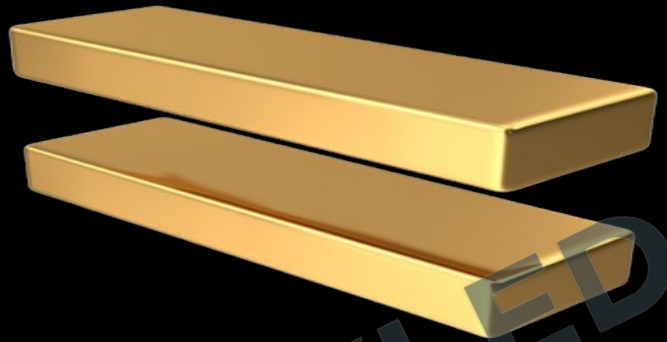
KG Placement Prep

Sanchit Socket

4. Operators, If-else & Number System

1. Assignment Operator
2. Arithmetic Operators
3. Order of Operation
4. Shorthand Operators
5. Unary Operators
6. If-else
7. Relational Operators
8. Logical Operators
9. Operator Precedence
10. Intro to Number System
11. Intro to Bitwise Operators

4.1 Assignment Operator



Assigns the right-hand operand's value to the left-hand operand.

Example: int a = 5;.

7. Create a program to swap two numbers.

4.2 Arithmetic Operators

Operators	Meaning	Example	Result
+	Addition	4+2	6
-	Subtraction	4-2	2
*	Multiplication	4*2	8
/	Division	4/2	2
%	Modulus operator to get remainder in integer division	5%2	1

4.3 Order of Operation

B	O	D	M	A	S
Bracket	Order	Divide	Multiply	Add	Subtract
( )	$\sqrt{x}$ x^2	÷ or x	+ or -		
Parentheses	Exponents	Multiply	Divide	Add	Subtract
P	E	M	D	A	S

4.4 Shorthand Operators

Operator symbol	Name of the operator	Example	Equivalent construct
+=	Addition assignment	x += 4;	x = x + 4;
-=	Subtraction assignment	x -= 4;	x = x - 4;
*=	Multiplication assignment	x *= 4;	x = x * 4;
/=	Division assignment	x /= 4;	x = x / 4;
%=	Remainder assignment	x %= 4;	x = x % 4;

4.5 Unary Operators

Operator	Description	Example
-	Converts a positive value to a negative	x = -y
Pre Increment	Increment the value by 1 and then use it in our statement	x = ++y
Pre Decrement	Decrement the value by 1 and then use it in our statement	x = --y
Post Increment	Use current value in the statement and then increment it by 1	x = y++
Post Decrement	Use current value in the statement and then decrement it by 1	x = y--

CHALLENGE

8. Create a program that takes two numbers and shows result of all arithmetic operators (+,-,*,/,%).
9. Create a program to calculate product of two floating points numbers.
10. Create a program to calculate Perimeter of a rectangle.
    Perimeter of rectangle ABCD = A+B+C+D
11. Create a program to calculate the Area of a Triangle.
    Area of triangle = ½*B*H
12. Create a program to calculate simple interest.
    Simple Interest = (P x T x R)/100
13. Create a program to calculate Compound interest.
    Compound Interest = P(1 + R/100)t
14. Create a program to convert Fahrenheit to Celsius
    °C = (°F - 32) × 5/9

4.6 if-else

1. Syntax: Uses if () {} to check a condition.
2. What is if: Executes block if condition is true, skips if false.
3. What is else: Executes a block when the if condition is false.
4. Curly Braces can be omitted for single statements, but not recommended.
5. If-else Ladder: Multiple if and else if blocks; only one executes.
6. Use Variables: Can store conditions in variables for use in if statements.

4.7 Relational Operators

< , > , <= , >= , == , !=

Equality

- == Checks value equality.

Inequality

- != Checks value inequality.

Relational

- > Greater than.
- < Less than.
- >= Greater than or equal to.
- <= Less than or equal to.

Order of Relational operators is less than arithmetic operators

4.8 Logical Operators


AND


Or

not

1. Types: && (AND), || (OR), ! (NOT)
2. AND (&&): All conditions must be true for the result to be true.
3. OR (||): Only one condition must be true for the result to be true.
4. NOT (!): Inverts the Boolean value of a condition.
5. Lower Priority than Math and Comparison operators

CHALLENGE

15.Create a program that determines if a number is positive, negative, or zero.

16.Create a program that determines if a number is odd or even.

17. Create a program that determines the greatest of the three numbers.

18.Create a program that determines if a given year is a leap year (considering conditions like divisible by 4 but not 100, unless also divisible by 400).

19.Create a program that calculates grades based on marks

A -> above 90% B -> above 75%

C -> above 60% D -> above 30%

F -> below 30%

20.Create a program that categorize a person into different age groups

Child -> below 13 Teen -> below 20

Adult -> below 60 Senior-> above 60

4.9 Operator Precedence

Operator Type	Category	Precedence	Associativity
Unary	postfix	a++, a--	Right to left
	prefix	++a, --a, +a, -a, ~, !	Right to left
Arithmetic	Multiplication	*, /, %	Left to Right
	Addition	+, -	Left to Right
Shift	Shift	<<, >>, >>>	Left to Right
Relational	Comparison	<, >, <=, >=, instanceOf	Left to Right
	equality	==, !=	Left to Right
Bitwise	Bitwise AND	&	Left to Right
	Bitwise exclusive OR	^	Left to Right
	Bitwise inclusive OR	|	Left to Right
Logical	Logical AND	&&	Left to Right
	Logical OR	||	Left to Right
Ternary	Ternary	? :	Right to Left
Assignment	assignment	=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=	Right to Left

Operator Precedence: Determines the evaluation order of operators in an expression based on their priority levels.

Associativity: Defines the order of operation for operators with the same precedence, usually left-to-right or right-to-left.

4.10 Intro to Number System

Number System	Base or Radix	Digits or symbols
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Number System	Base or Radix	Digits or symbols
Unary	1	0/1
Decimal	10	0,1,2,3,4,5,6,7,8,9

4.10 Intro to Number System

Decimal Number System

- Decimal Number System is a base-10 system that has ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- The decimal number system is said to be of base, or radix, 10 because it uses 10 digits and the coefficients are multiplied by powers of 10.
- This is the base that we often use in our day to day life.
- Example: $(7,392)_{10}$, where 7,392 is a shorthand notation for what should be written as

$$7 * 10^3 + 3 * 10^2 + 9 * 10^1 + 2 * 10^0$$

4.10 Intro to Number System

Binary Number System

- The coefficients of the binary number system have only two possible values: 0 and 1.
- Each coefficient a_j is multiplied by a power of the radix, e.g., 2^j, and the results are added to obtain the decimal equivalent of the number.

Example: $(11010.11)_2$ value in Decimal?

$$1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 + 1 * 2^{-1} + 1 * 2^{-2} = 26.75$$

4.10 Intro to Number System

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

4.11 Intro to Bitwise Operators

1.AND Operator (&): Performs on two integers. Each bit of the output is 1 if the corresponding bits of both operands are 1, otherwise 0.
2.OR Operator (|): Performs on two integers. Each bit of the output is 0 if the corresponding bits of both operands are 0, otherwise 1.
3.XOR Operator (^): Performs on two integers. Each bit of the output is 1 if the corresponding bits of the operands are different.
4.NOT Operator (~): Performs a bitwise complement. It inverts the bits of its operand (0 becomes 1, and 1 becomes 0).
5.Left Shift Operator (<<): Shifts the left operand's bits to the left by the number of positions specified by the right operand, filling the new rightmost bits with zeros.
6.Right Shift Operator (>>): Shifts the left operand's bits to the right. If the left operand is positive, zeros are filled into the new leftmost bits; if negative, ones are filled in.

CHALLENGE

21. Create a program that shows bitwise AND of two numbers.
22. Create a program that shows bitwise OR of two numbers.
23. Create a program that shows bitwise XOR of two numbers.
24. Create a program that shows bitwise compliment of a number.
25. Create a program that shows use of left shift operator.
26. Create a program that shows use of right shift operator.
27. Write a program to check if a given number is even or odd using bitwise operators.

Revision

1. Assignment Operator
2. Arithmetic Operators
3. Order of Operation
4. Shorthand Operators
5. Unary Operators
6. If-else
7. Relational Operators
8. Logical Operators
9. Operator Precedence
10. Intro to Number System
11. Intro to Bitwise Operators

Practice Exercise

Java Basics

Answer in True/False:

1.In Java, && and || operators perform short-circuit evaluation.
2.In an if-else statement, the else block executes only when the if condition is false.
3.Java allows an if statement without the else part.
4.The ^ operator in Java is used for exponentiation.
5.Unary minus operator can be used to negate the value of a variable in Java.
6.a += b is equivalent to a = a + b in Java.
7.In Java, the binary number system uses base 10.
8.The number 1010 in binary is equivalent to 10 in decimal.
9.& and | are logical operators in Java.
10.In Java, a >> 2 shifts the binary bits of a to the left by 2 positions.

Practice Exercise

Java Basics

Answer in True/False:

1. In Java, && and || operators perform short-circuit evaluation. — True
2. In an if-else statement, the else block executes only when the if condition is false. — True
3. Java allows an if statement without the else part. — True
4. The ^ operator in Java is used for exponentiation. — False
5. Unary minus operator can be used to negate the value of a variable in Java. — True
6. a += b is equivalent to a = a + b in Java. — True
7. In Java, the binary number system uses base 10. — False
8. The number 1010 in binary is equivalent to 10 in decimal. — True
9. & and | are logical operators in Java. — False
10. In Java, a >> 2 shifts the binary bits of a to the left by 2 positions. — False

KG Coding

Some Other One shot Video Links:

- Complete HTML
- Complete CSS
- Complete JavaScript
- Complete React and Redux
- One shot University Exam Series

http://www.kgcoding.in/

Our YouTube Channels

KG Coding Android App

KG Coding

Knowledge GATE

KG Placement Prep

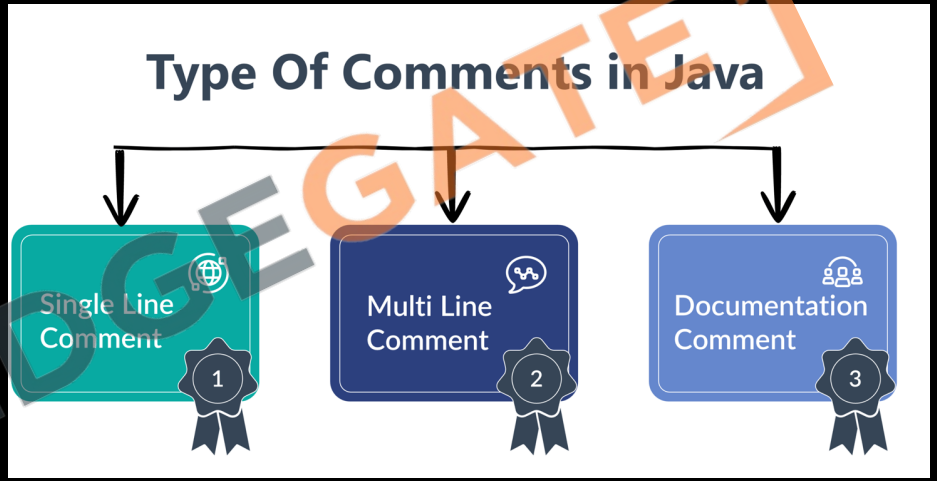
Sanchit Socket

5 While Loop, Methods & Arrays

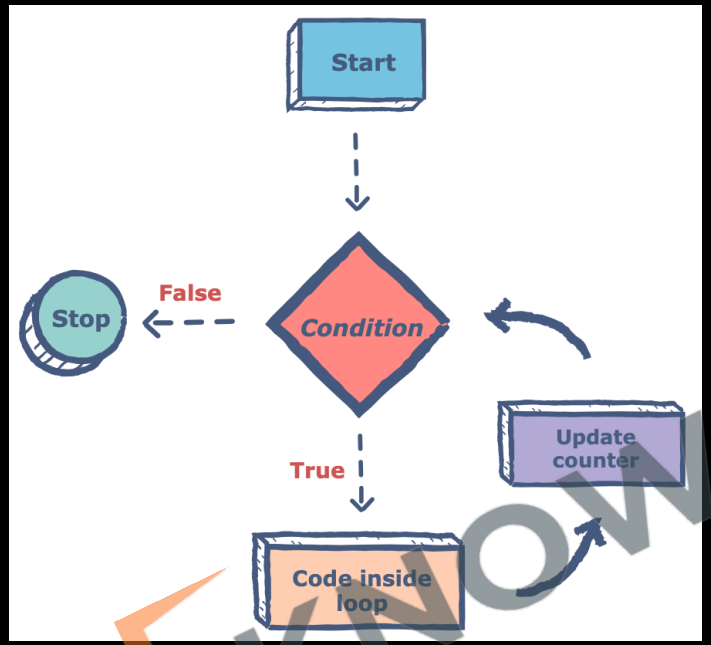
1. Comments
2. While Loop
3. Methods
4. Return statement
5. Arguments
6. Arrays
7. 2D Arrays

5.1 Java Comments

1. Used to add notes in Java code
2. Not displayed on the web page
3. Helpful for code organization
4. Syntax:
    1. Single Line: //
    2. Multi Line: /* */
    3. Java Docs: /** */

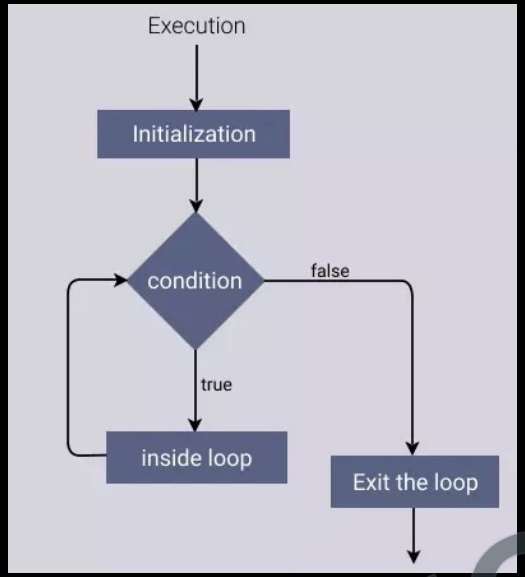


5.2 What is a Loop?



1. Code that runs multiple times based on a condition.
2. Repeated execution of code.
3. Loops automate repetitive tasks.
4. Types of Loops: while, for, while, do-while.
5. Iterations: Number of times the loop runs.

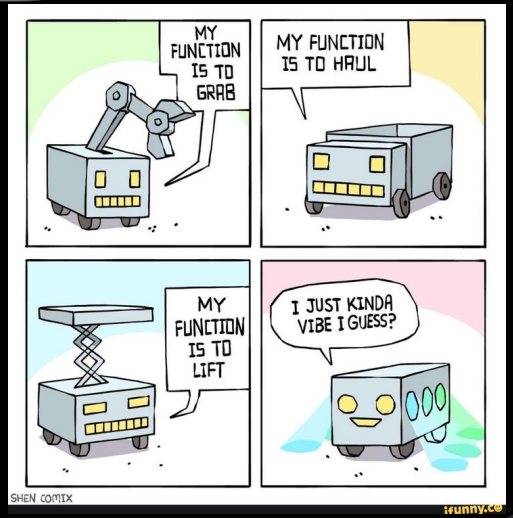
5.2 While Loop




```
while (condition) {
    // Body of the loop
}
```

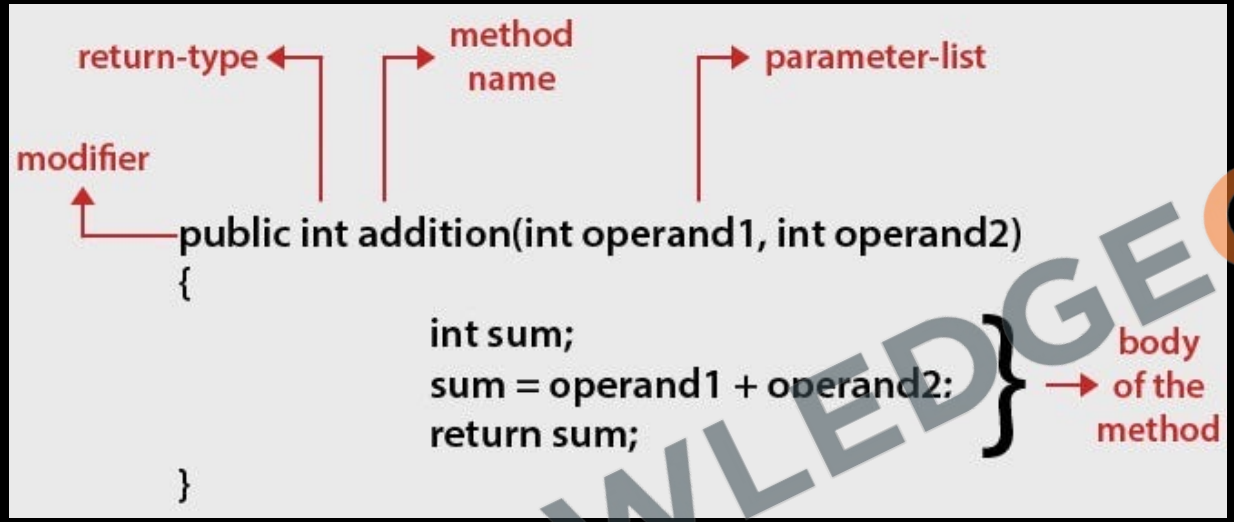

1. Iterations: Number of times the loop runs.
2. Used for non-standard conditions.
3. Repeating a block of code while a condition is true.
4. Remember: Always include an update to avoid infinite loops.

5.3 What are Functions / Methods?



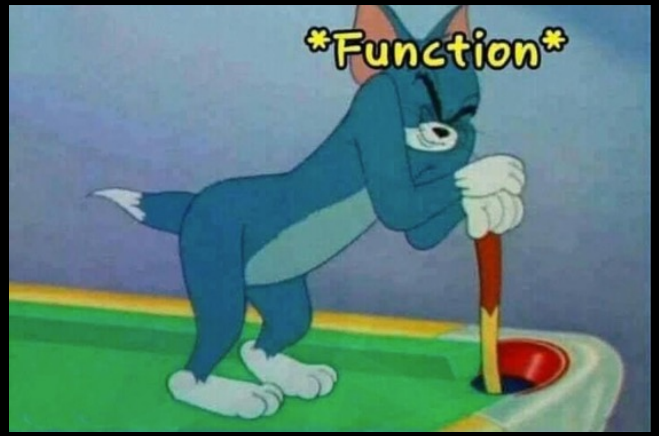
1. Definition: Blocks of reusable code.
2. DRY Principle: "Don't Repeat Yourself" it Encourages code reusability.
3. Usage: Organizes code and performs specific tasks.
4. Naming Rules: Same as variable names: camelCase
5. Example: ”Beta Gas band kar de"

5.3 Method Syntax



1. Follows same rules as variable names.
2. Use () to contain parameters.
3. Invoke by using the function name followed by ().
4. Fundamental for code organization and reusability.

5.4 Return statement



1. Sends a value back from a function.
2. Example: "Ek glass paani laao"
3. What Can Be Returned: Value, variable, calculation, etc.
4. Return ends the function immediately.
5. Function calls make code jump around.
6. Prefer returning values over using global variables.

5.5 Arguments vs Parameters

Parameter

Function

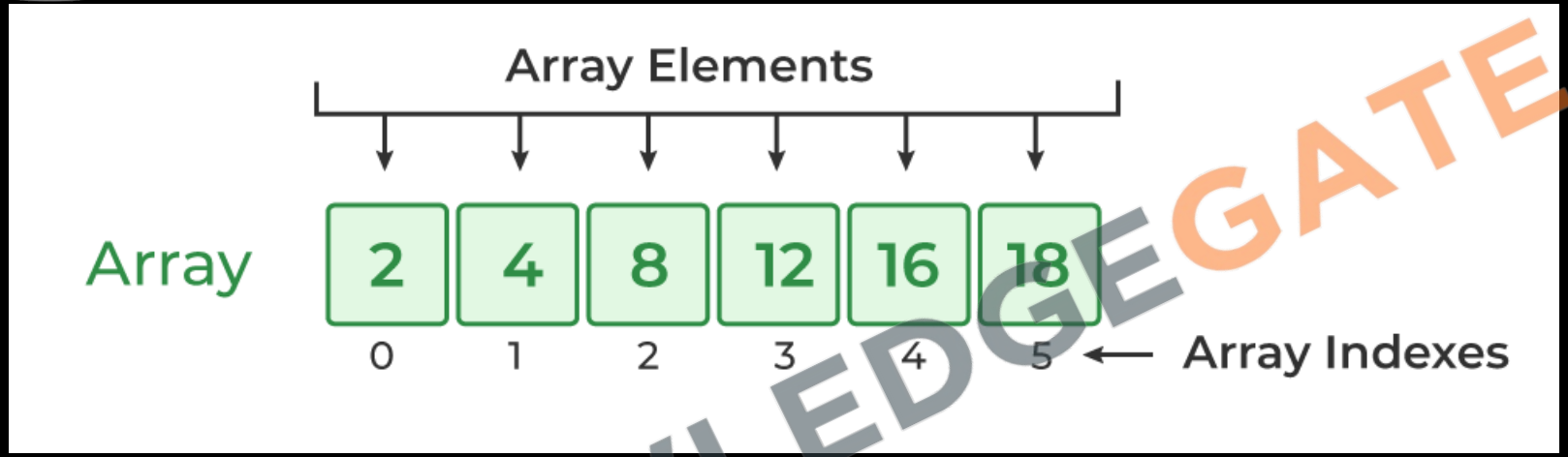
Return

1. Input values that a function takes.
2. Parameters put value into function, while return gets value out.
3. Example: "Ek packet dahi laao"
4. Naming Convention: Same as variable names.
5. Parameter vs Argument
6. Examples: System.out.print, Scanner.nextInt(), are functions we have already used
7. Multiple Parameters: Functions can take more than one.
8. Default Value: Can set a default value for a parameter.

CHALLENGE

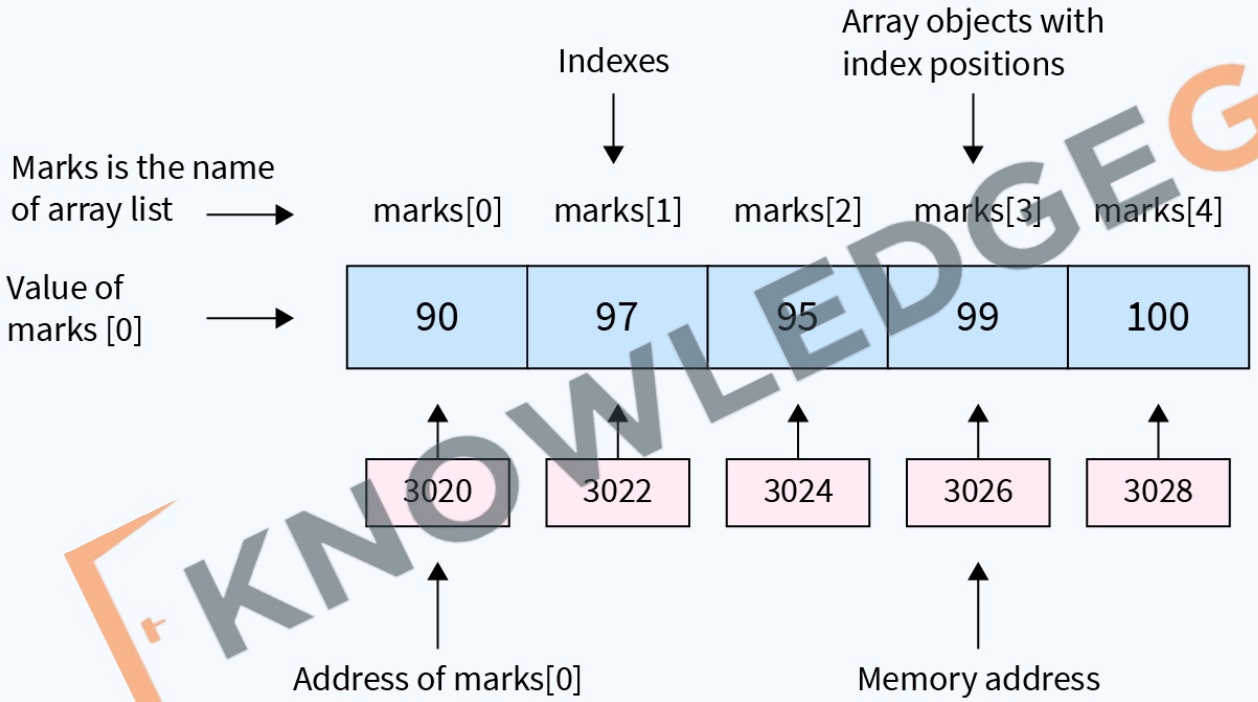
28. Develop a program that prints the multiplication table for a given number.
29. Create a program to sum all odd numbers from 1 to a specified number N.
30. Write a function that calculates the factorial of a given number.
31. Create a program that computes the sum of the digits of an integer.
32. Create a program to find the Least Common Multiple (LCM) of two numbers.
33. Create a program to find the Greatest Common Divisor (GCD) of two integers.
34. Create a program to check whether a given number is prime.
35. Create a program to reverse the digits of a number.
36. Create a program to print the Fibonacci series up to a certain number.
37. Create a program to check if a number is an Armstrong number.
38. Create a program to verify if a number is a palindrome.
39. Create a program that print patterns:

5.6 What is an Array?

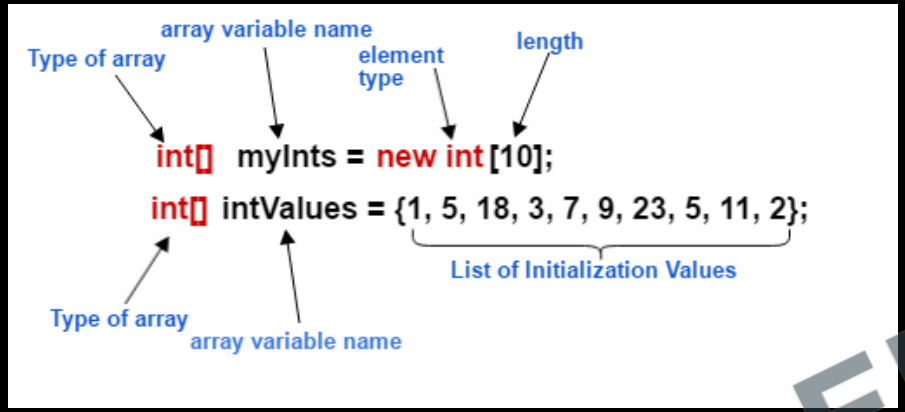


1. An Array is just a list of values.
2. Index: Starts with 0.
3. Arrays are used for storing multiple values in a single variable.

5.6 Array Memory



5.6 Array Syntax



1. Use {} to create a new array, {} brackets enclose list of values
2. Arrays can be saved to a variable.
3. Accessing Values: Use [] with index.
4. Syntax Rules:
   - Brackets start and end the array.
   - Values separated by commas.
   - Can span multiple lines.

5.6 Array Length

Array length = Array's Last Index + 1

Array's last index = 7

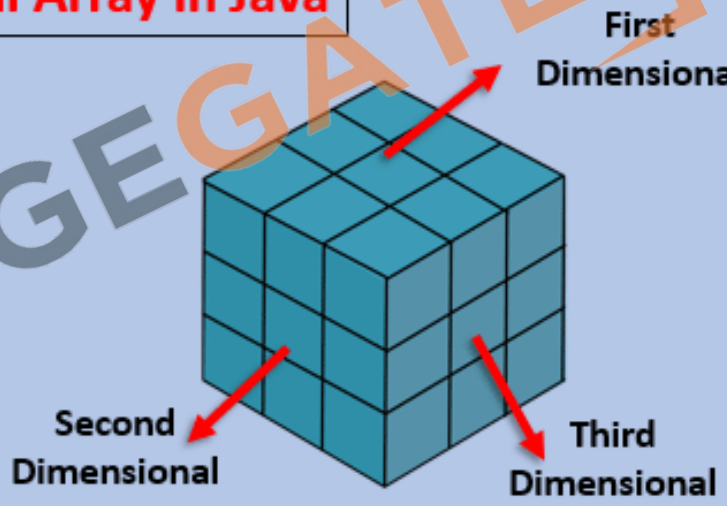
Arraylength = 7 + 1 = 8

5.7 2D Arrays

Types of Multidimensional Array in Java

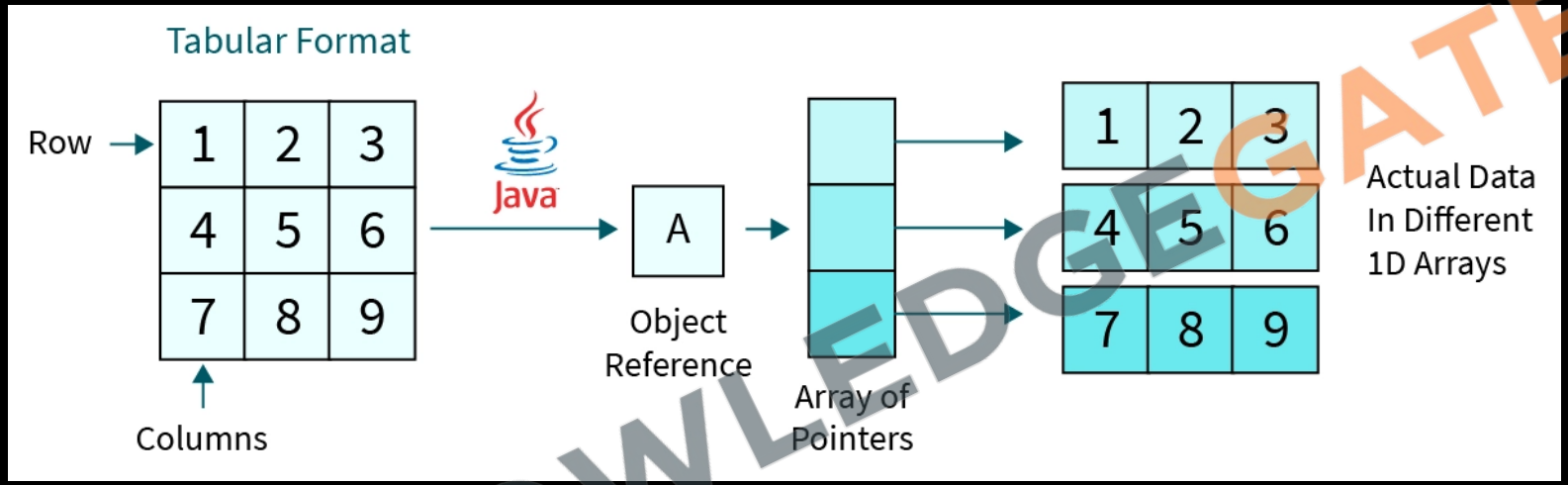
	Column 0	Column 1	Column 2
Row 0	X[0][0]	X[0][1]	X[0][2]
Row 1	X[1][0]	X[1][1]	X[1][2]
Row 2	X[2][0]	X[2][1]	X[2][2]

2D-Array



3D-Array

5.7 2D Arrays



5.7 2D Arrays

```
int[][] numArr = new int[2][3];
int[][] inArray = {{1, 2, 5}, {8, 9, 4}};

numArr[0][0] = 5;
```

CHALLENGE

40. Create a program to find the sum and average of all elements in an array.
41. Create a program to find number of occurrences of an element in an array.
42. Create a program to find the maximum and minimum element in an array.
43. Create a program to check if the given array is sorted.
44. Create a program to return a new array deleting a specific element.
45. Create a program to reverse an array.
46. Create a program to check is the array is palindrome or not.
47. Create a program to merge two sorted arrays.
48. Create a program to search an element in a 2-D array.
49. Create a program to do sum and average of all elements in a 2-D array.
50. Create a program to find the sum of two diagonal elements.

Revision

1. Comments
2. While Loop
3. Methods
4. Return statement
5. Arguments
6. Arrays
7. 2D Arrays

Practice Exercise

Java Basics

Answer in True/False:

1. A 'while' loop in Java continues to execute as long as its condition is true.
2. The body of a 'while' loop will execute at least once, regardless of the condition.
3. A 'while' loop cannot be used for iterating over an array.
4. Infinite loops are not possible with 'while' loops in Java.
5. A method in Java can return more than one value at a time.
6. It's mandatory for every Java method to have a return type.
7. The size of an array in Java can be modified after it is created.
8. Arrays in Java can contain elements of different data types.
9. An array is a reference type in Java.
10. Java arrays are zero-indexed, meaning the first element has an index of 0.

Practice Exercise

Java Basics

Answer in True/False:

1. A 'while' loop in Java continues to execute as long as its condition is true. — True
2. The body of a 'while' loop will execute at least once, regardless of the condition. — False
3. A 'while' loop cannot be used for iterating over an array. — False
4. Infinite loops are not possible with 'while' loops in Java. — False
5. A method in Java can return more than one value at a time. — False
6. It's mandatory for every Java method to have a return type. — True
7. The size of an array in Java can be modified after it is created. — False
8. Arrays in Java can contain elements of different data types. — False
9. An array is a reference type in Java. — True
10. Java arrays are zero-indexed, meaning the first element has an index of 0. — True

KG Coding

Some Other One shot Video Links:

- Complete HTML
- Complete CSS
- Complete JavaScript
- Complete React and Redux
- One shot University Exam Series

http://www.kgcoding.in/

Our YouTube Channels

KG Coding Android App

KG Coding

Knowledge GATE

KG Placement Prep

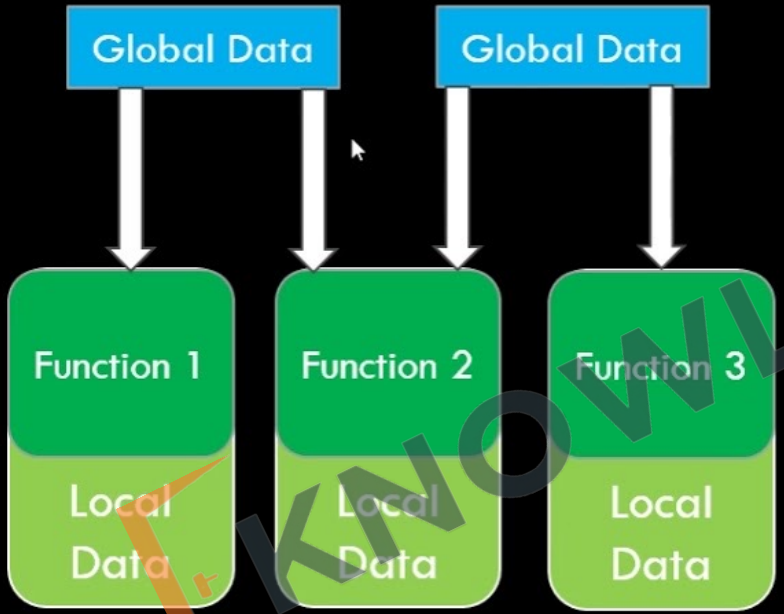
Sanchit Socket

6 Classes & Objects

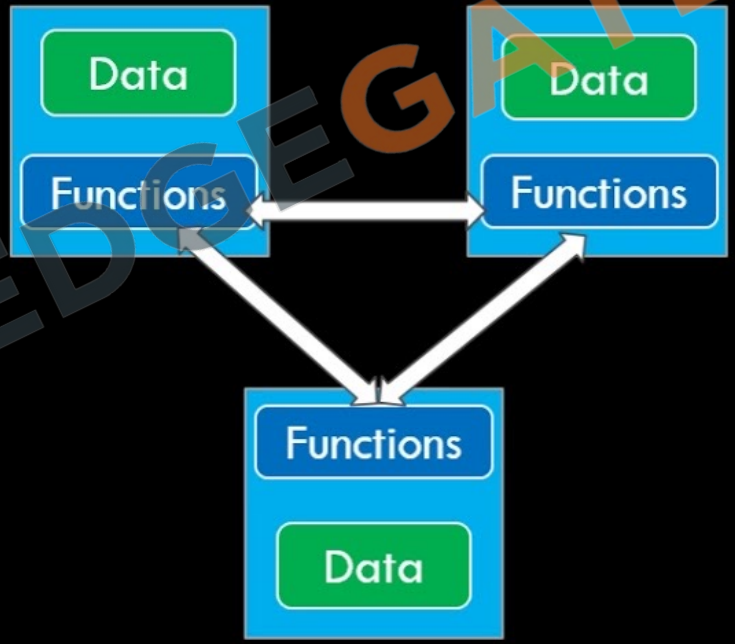
1. Process vs Object Oriented
2. Instance Variables and Methods
3. Declaring and Using Objects
4. Class vs Object
5. This & Static Keyword
6. Constructors & Code Blocks
7. Stack vs Heap Memory
8. Primitive vs Reference Types
9. Variable Scopes
10. Garbage Collection & Finalize

6.1 Process vs Object Oriented

Procedural Oriented Programming



Object Oriented Programming



6.1 Process vs Object Oriented

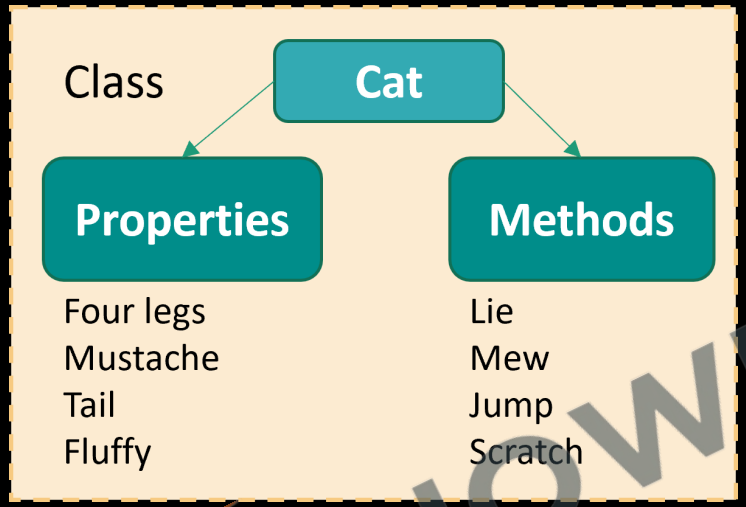
- Procedural

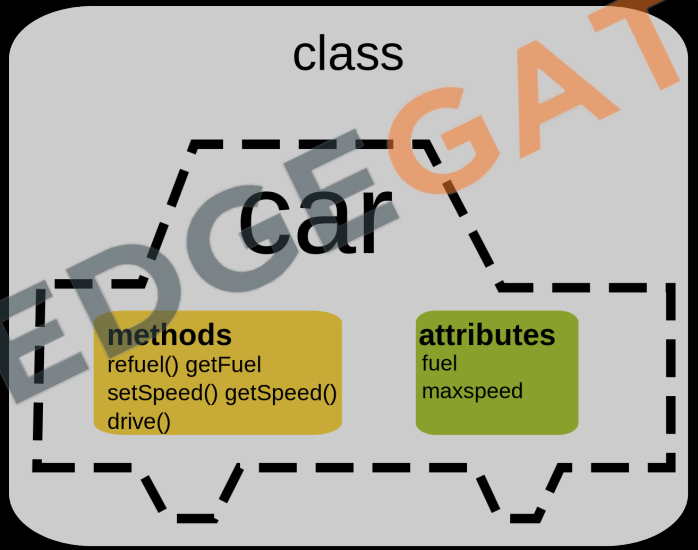
Withdraw, deposit, transfer

- Object Oriented

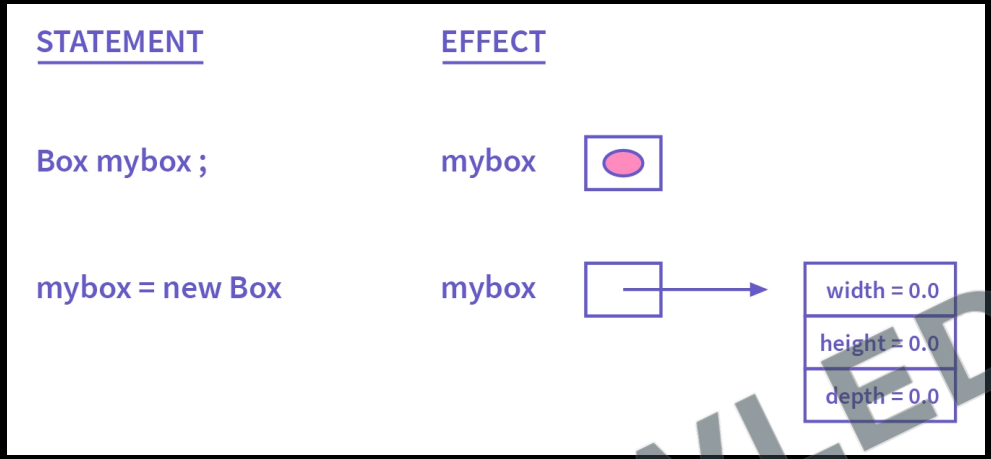
Customer, money, account

6.2 Instance Variables and Methods



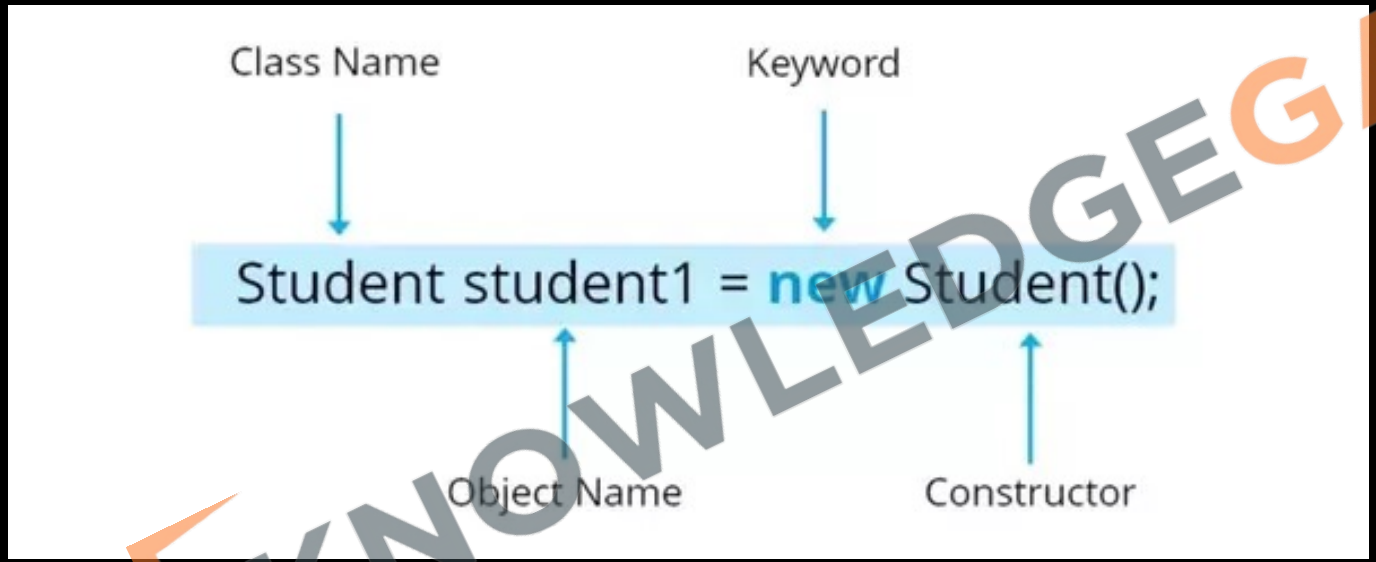


6.3 Declaring Objects

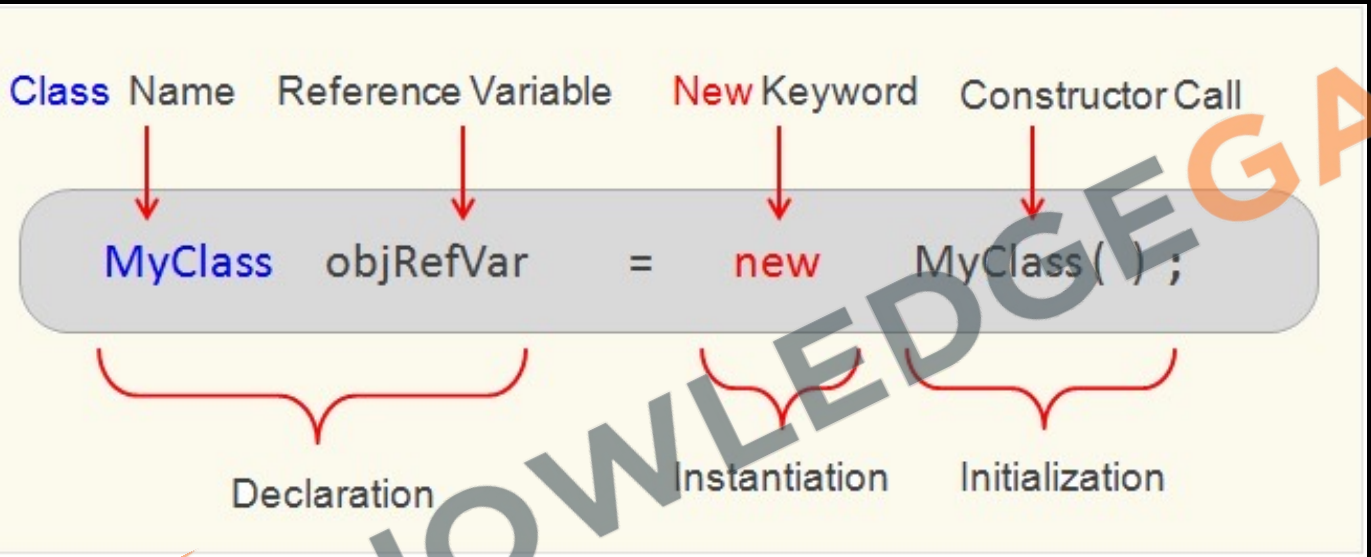


1. Object Creation: new instantiates a new object of a class.
2. Memory Allocation: Allocates memory for the object in the heap.
3. Constructor Invocation: Calls the class constructor to initialize the object.
4. Reference Return: Returns a reference to the created object.
5. Array Creation: Also used for creating arrays, like int[] arr = new int[5];.
6. Dynamic Allocation: Unlike static allocation, new allows for dynamic memory allocation, allocating memory at runtime.

6.3 Declaring Objects Syntax



6.3 Declaring Objects Syntax



objRefVar is a Reference Variable of MyClass Type .

new is a Java Keyword used To Instantiate a Class .

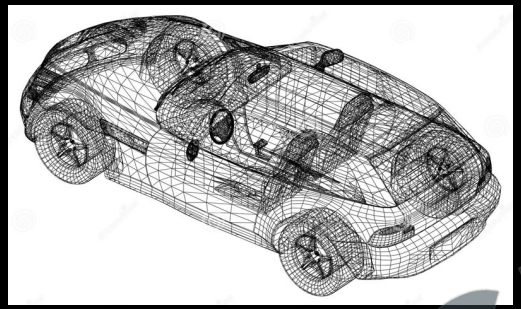
6.3 Using Objects

Access properties using . Operator like product.price

6.4 Class vs Object

Class is a blueprint; Objects are real values in memory.

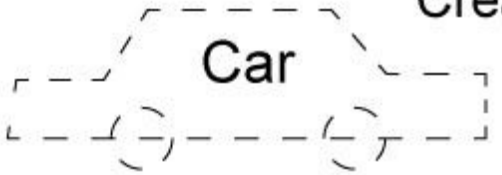
6.4 Class vs Object



Class is a blueprint; Objects are real values in memory.

6.4 Class vs Object

Class



Properties
color
price
km
model

Methods - behaviors
start()
backward()
forward()
stop()

Create an instance

Object

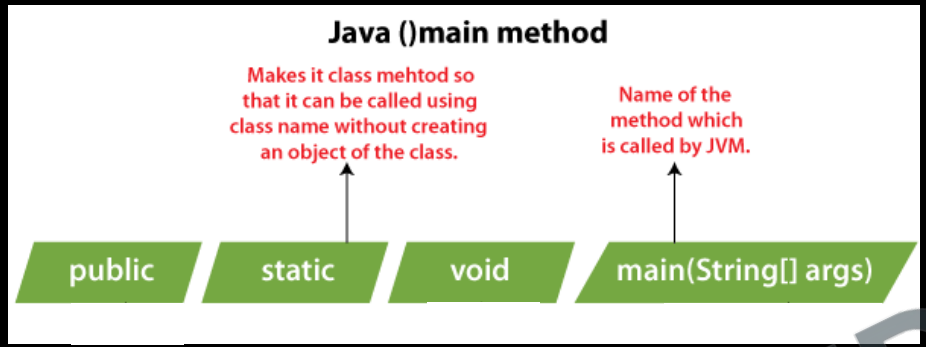
Property values
color: red
price: 23,000
km: 1,200
model: Audi

Methods
start()
backward()
forward()
stop()

6.5 This Keyword

1. Current Instance: Refers to the current class instance variable.
2. Constructor Call: Can be used to invoke a constructor of the same class (this()).
3. Method Call: Invokes a method of the current object.
4. Pass as Argument: Can be passed as an argument in the method call.
5. Return the Current Class Instance: Can return the current class instance from the method.

6.5 Static Keyword



1. Static Variables: Belong to the class, not individual instances. Shared among all instances of the class.
2. Static Methods: Can be called without creating an object of the class. Can only directly access static variables and other static methods.
3. No Access to Non-static Members: Static methods and blocks cannot directly access non-static members (variables and methods) of the class.

6.6 Constructors

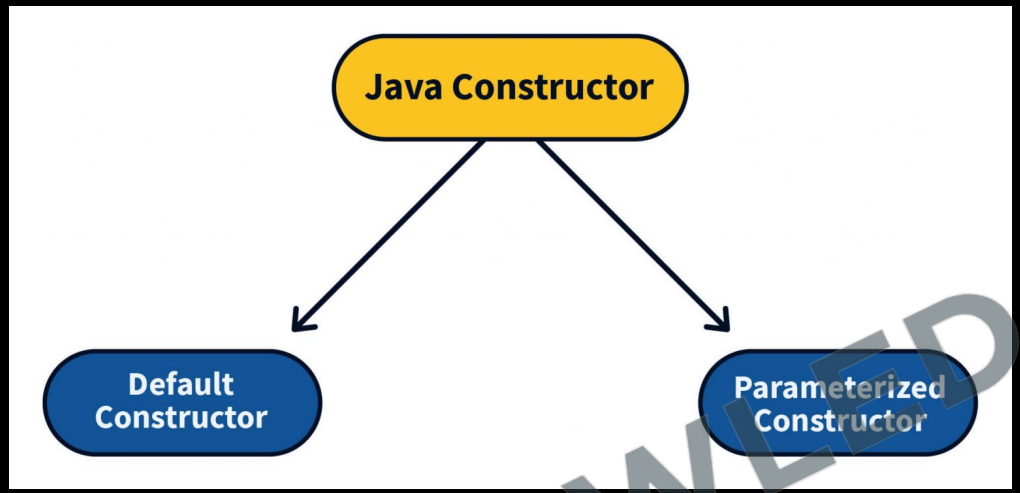
```java
public class Car {
    1 usage
    String color;
    1 usage
    float price;

    1 usage   new *
    Car() { // Default constructor
        color = "Black";
        price = 50000;
    }

    new *
    public static void main(String[] args) {
        Car swift = new Car();
    }
}
```

1. Purpose: Constructors initialize new objects and set initial states for the object's attributes.
2. Naming: A constructor must have the same name as the class in which it is declared.
3. No Return Type: Constructors do not have a return type, not even void.
4. Automatic Calls: A constructor is automatically called when an instance of a class is created.

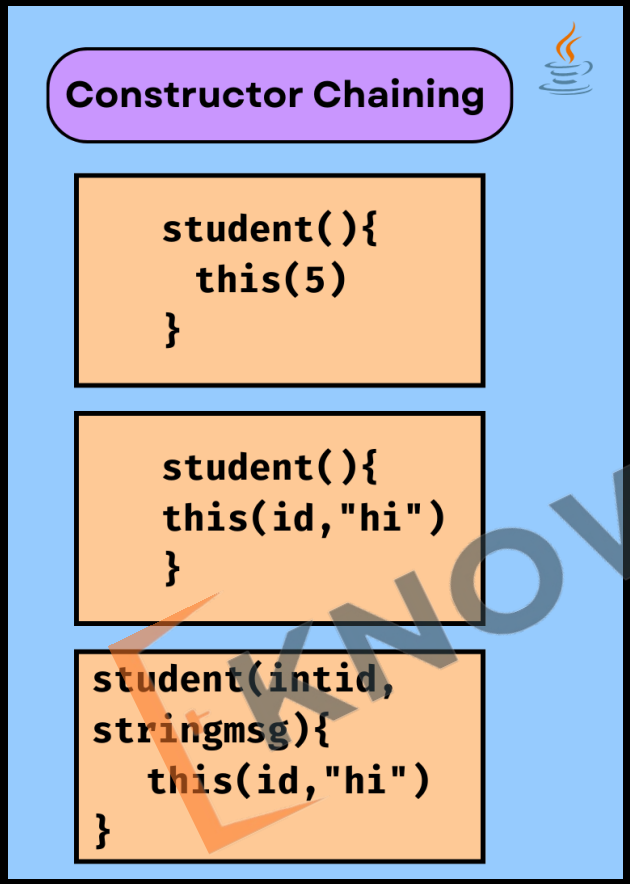
6.6 Constructors (Types)



1. Default Constructor: If no constructor is explicitly defined, Java provides a default constructor that initializes all member variables to default values.
2. Parameterized Constructors: Constructors can have parameters to pass values when creating an object, allowing for different initializations.

```
Car(String carColor, float currPrice) {
    color = carColor;
    price = currPrice;
}
```

6.6 Constructors (Chaining)



1. Within Same Class: Using this() to call another constructor in the same class.
2. First Statement: this() must be the first statement in a constructor.
3. No Loop: Constructor chaining can't form a loop; it must have a termination point.

6.6 Code Blocks

```
{
    System.out.println("This is a Initialization Block");
    color = "Black";
    price = 50000;
}
```

```
static {
    System.out.println("This is a Static Block");
}
```

```
if (true) { // code block
    System.out.println("Code Block");
}
```


1. Scope: Code blocks {} determine the scope of variables.
2. Local Variables: Variables inside a block are not accessible outside it.
3. Initialization Block: Blocks without static run each time an instance is created.
4. Static Block: Blocks with static run once when the class is loaded.

CHALLENGE

51. Create a Book class for a library system.
    - Instance variables: title, author, isbn.
    - Static variable: totalBooks, a counter for the total number of book instances.
    - Instance methods: borrowBook(), returnBook().
    - Static method: getTotalBooks(), to get the total number of books in the library.
52. Design a Course class.
    - Instance variables: courseName, enrolledStudents.
    - Static variable: maxCapacity, the maximum number of students for any course.
    - Instance methods: enrollStudent(String studentName), unenrollStudent(String studentName).
    - Static method: setMaxCapacity(int capacity), to set the maximum capacity for courses.

KG Coding

Some Other One shot Video Links:

- Complete HTML
- Complete CSS
- Complete JavaScript
- Complete React and Redux
- One shot University Exam Series

http://www.kgcoding.in/

Our YouTube Channels

KG Coding Android App

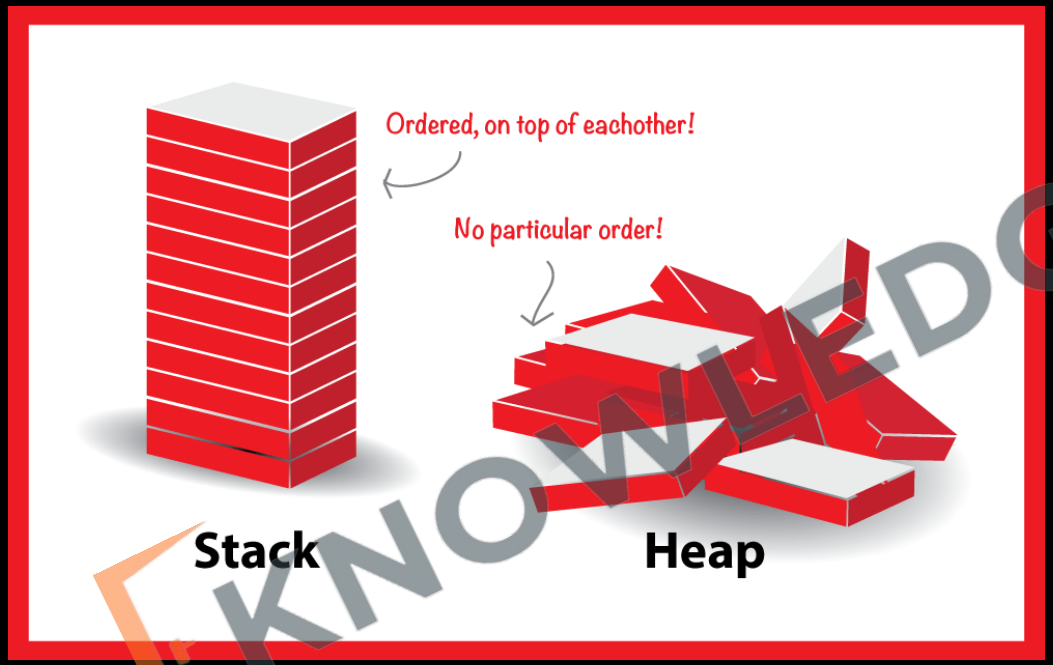
KG Coding

Knowledge GATE

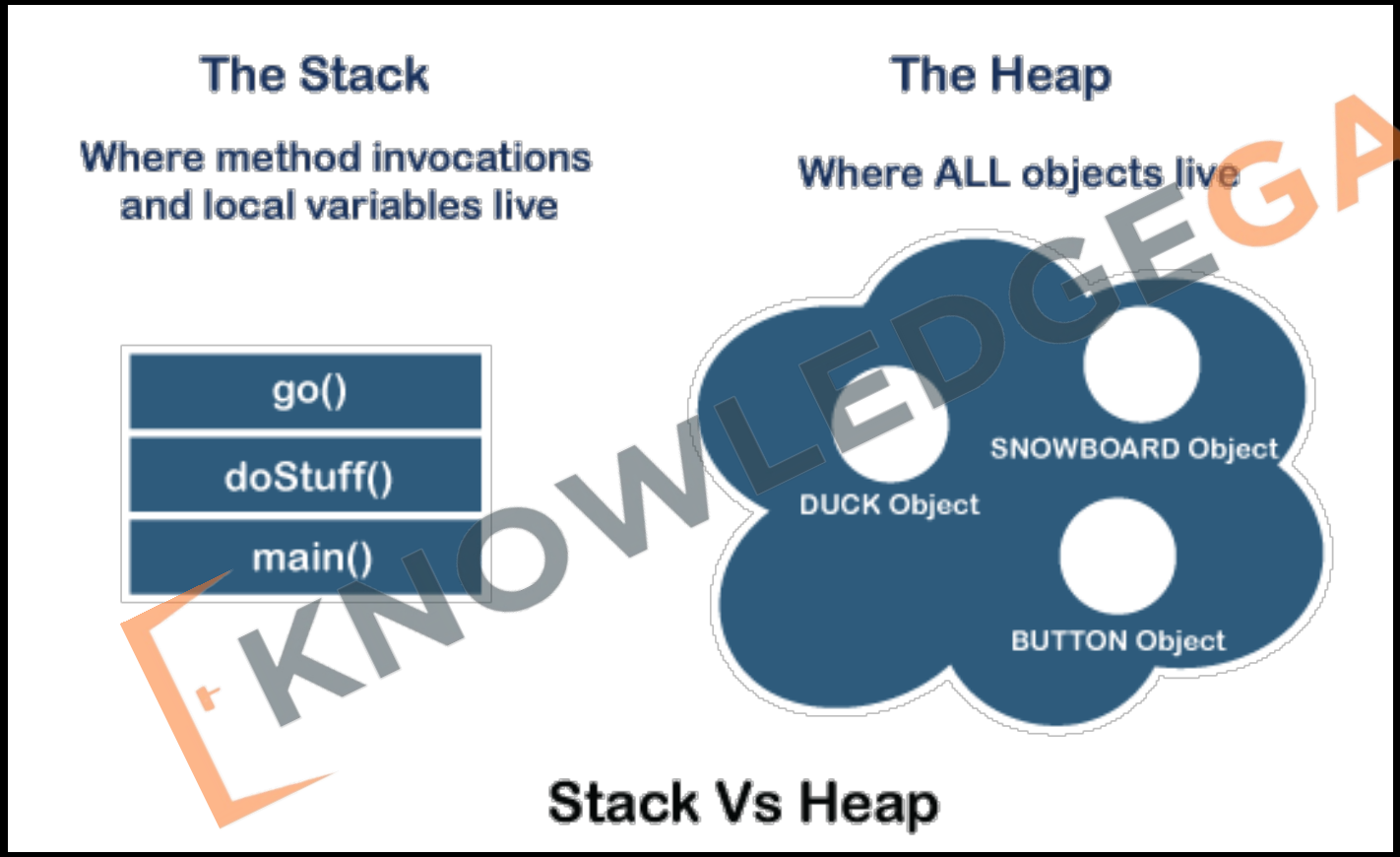
KG Placement Prep

Sanchit Socket

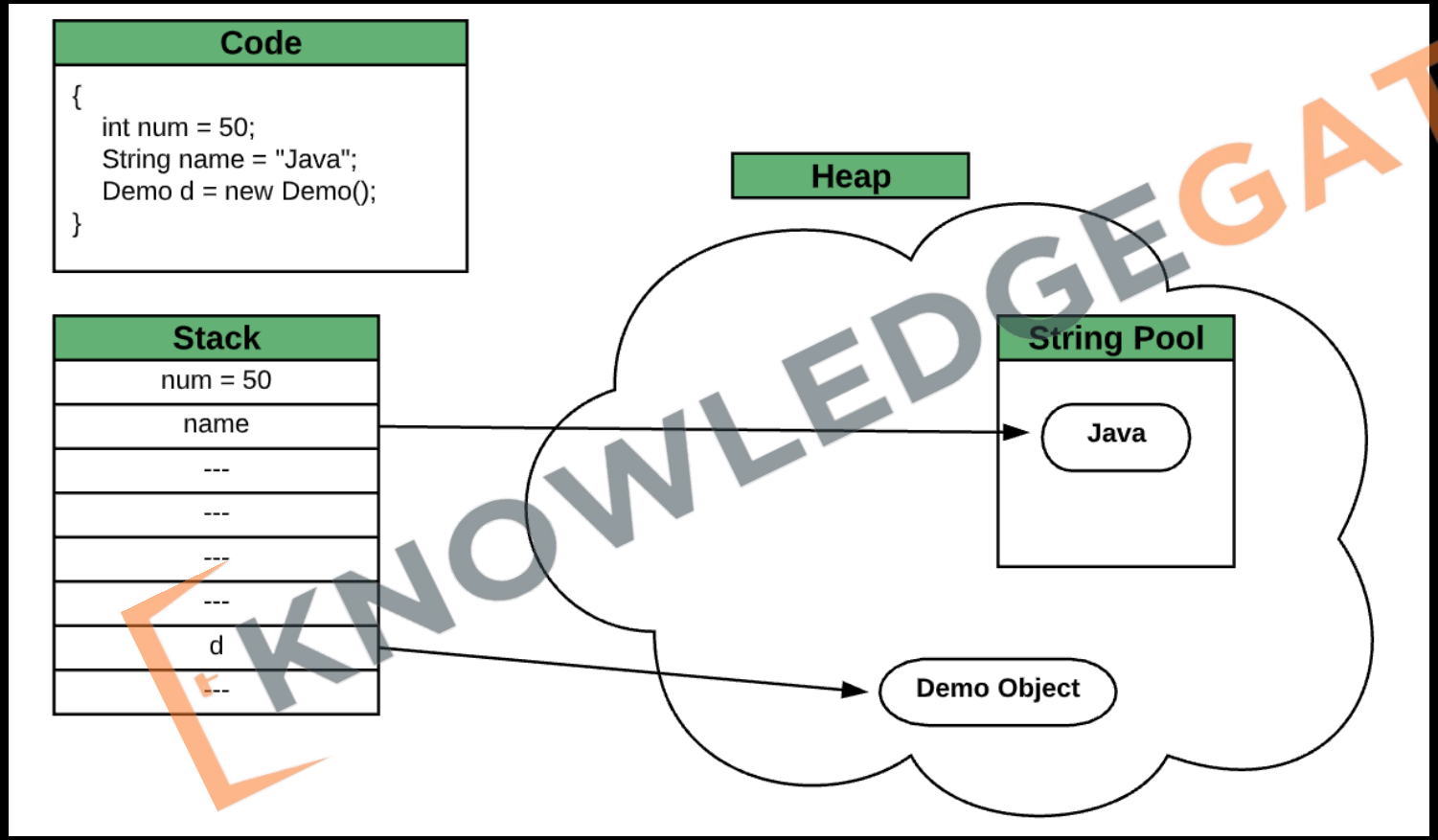
6.7 Stack vs Heap Memory



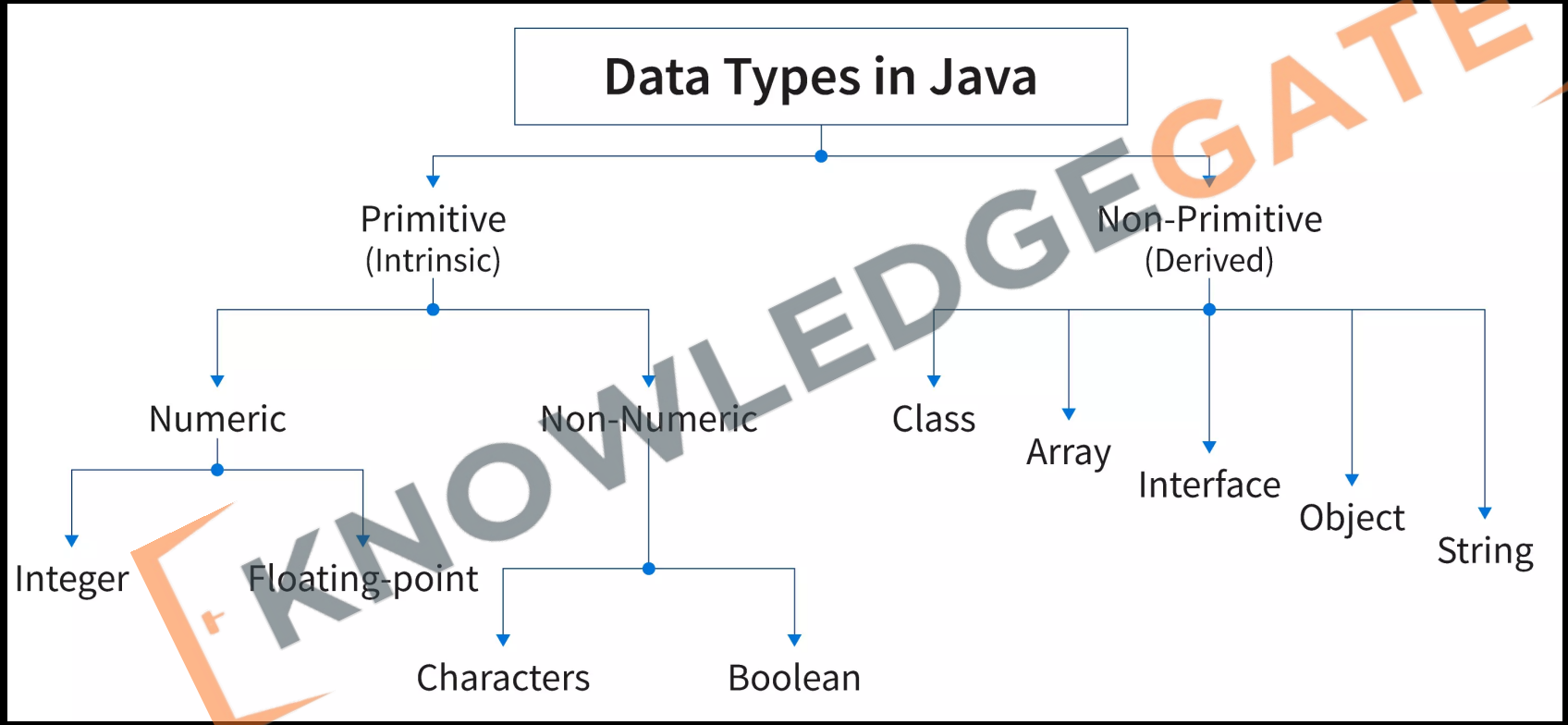
6.7 Stack vs Heap Memory



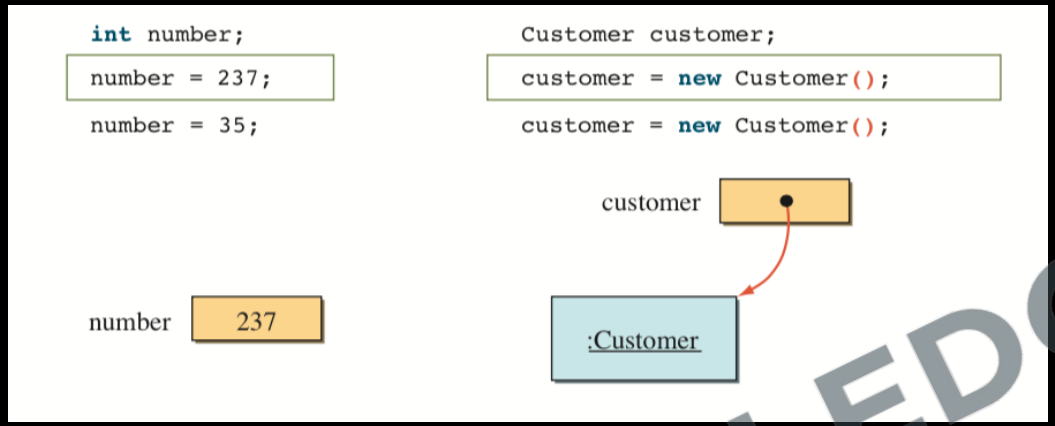
6.7 Stack vs Heap Memory



6.8 Primitive vs Reference Types



6.8 Primitive vs Reference Types



1. Memory: Primitives store actual values; reference types store addresses to objects.
2. Default Values: Primitives have specific defaults like 0 or false; reference types default to null.
3. Speed: Access to primitives is generally faster.
4. Storage Location: Primitives are stored in the stack; reference types are stored in the heap.
5. Comparison: Primitives compared by value; reference types compared by reference.

6.9 Variable Scopes

Instance vs Local Variables

```
class Athlete {
        public String   athleteName;
        public double   athleteSpeed;
        public int      athleteAge;

        public Athlete( name. speed, age ){
                this.athleteName = name;
                this.athleteSpeed = speed;
                this.atheleteAge = age;
        }
        public void athleteRun(){
                int speed = 100;
                System.out.println("Athlete runs at"+ speed +"Km/hr");
        }
}
```

Instance Variables

Local Variables

6.9 Variable Scopes

global variable

local variable

local variable

```
public class MyScopeExample {
    static String root = "I'm available
to all lines of code within my context";

public static void main(String[] args) {
    String spy = "I'm a spy";
    System.out.println(root); // Ok
    System.out.println(spy); // Ok
    System.out.println(anotherSpy); // Error
}

public static void helpfulFunction() {
    String anotherSpy = "I'm another spy";
    System.out.println(root); // Ok
    System.out.println(anotherSpy); // Ok
    System.out.println(spy); // Error
}
}
```

Local Scope

Local Scope

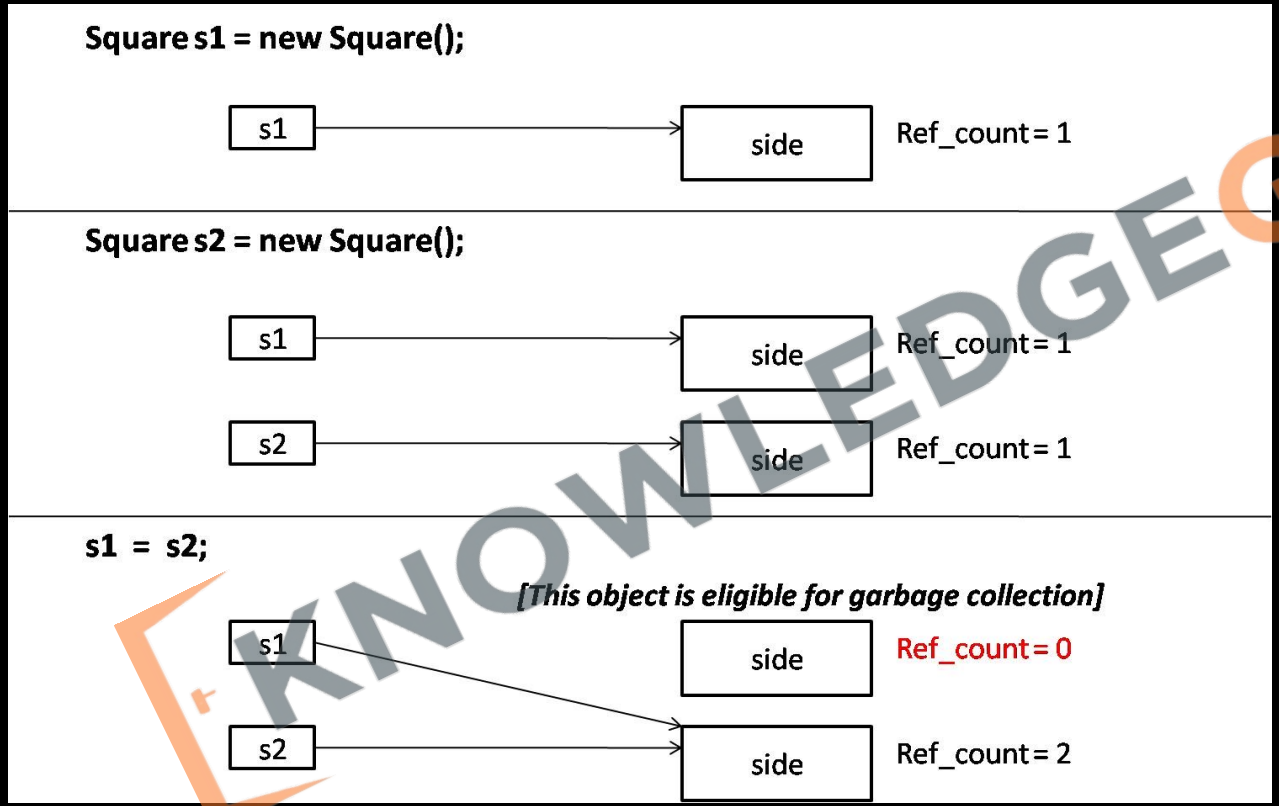
Global Scope

Multiple scopes

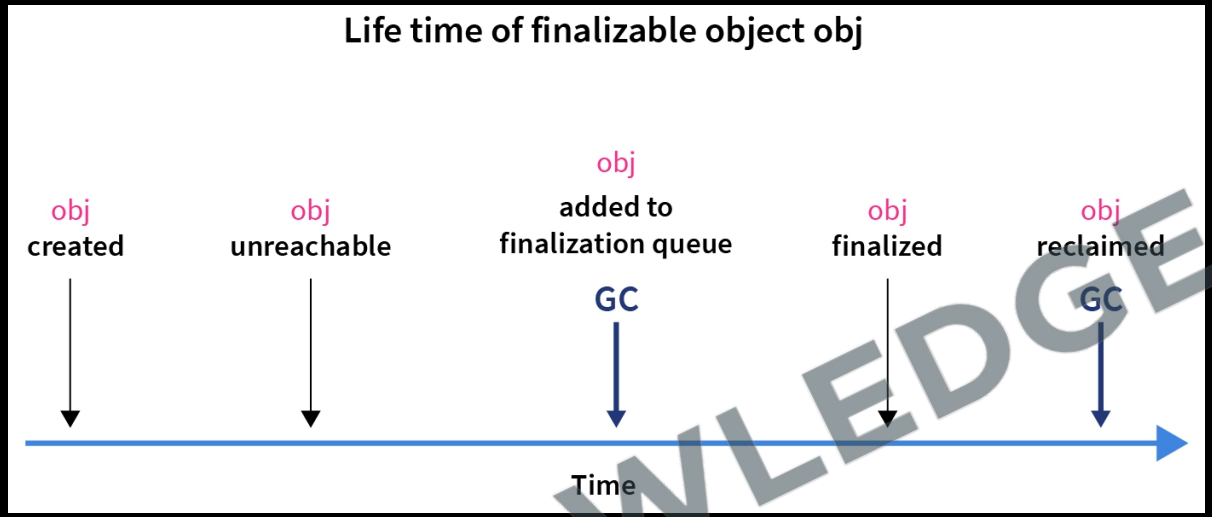
6.10 Garbage Collection & Finalize

1. Automatic Process: Garbage collection is managed by the Java Virtual Machine (JVM), running in the background.
2. Object Eligibility: Objects that are no longer reachable, meaning no active references to them, are eligible for garbage collection.
3. No Manual Control: Unlike languages like C++, Java developers cannot explicitly deallocate memory. Garbage collection is automatic and non-deterministic.
4. Generational Collection: Java uses a generational garbage collection strategy, which divides memory into different regions (young, old, and permanent generations) based on object ages.
5. Heap Memory: Garbage collection occurs in the heap memory, where all Java objects reside.
6. Performance Impact: Garbage collection can affect application performance, particularly if it runs frequently or takes a long time to complete.

6.10 Garbage Collection & Finalize



6.10 Garbage Collection & Finalize



1. Finalization: Before an object is garbage collected, the finalize() method may be called, giving the object a chance to clean up resources. However, it's not guaranteed to run, and its usage is generally discouraged.
2. Optimization: Developers can optimize the process indirectly through code practices, like setting unnecessary object references to null.
3. System.gc() Call: While System.gc() suggests that the JVM performs garbage collection, it's not a guarantee.

Revision

1. Process vs Object Oriented
2. Instance Variables and Methods
3. Declaring and Using Objects
4. Class vs Object
5. This & Static Keyword
6. Constructors & Code Blocks
7. Stack vs Heap Memory
8. Primitive vs Reference Types
9. Variable Scopes
10. Garbage Collection & Finalize

Practice Exercise

Classes, Objects

Answer in True/False:

1. Object-oriented programming is mainly concerned with data rather than logic.
2. Instance methods in Java can be called without creating an object of the class.
3. A class in Java is a template that can be used to create objects.
4. Static methods belong to the class and can be called without an object.
5. The new keyword is used to declare an array in Java.
6. Variables declared within a method are called instance variables.
7. In Java, you can access instance variables through static methods directly.
8. Every object in Java has its own unique set of instance variables.
9. The static keyword in Java means that a particular member belongs to a type itself, rather than to instances of that type.
10. Variable scope in Java is determined at runtime.

Practice Exercise

Classes, Objects

Answer in True/False:

1. Object-oriented programming is mainly concerned with data rather than logic. — True
2. Instance methods in Java can be called without creating an object of the class. — False
3. A class in Java is a template that can be used to create objects. — True
4. Static methods belong to the class and can be called without an object. — True
5. The new keyword is used to declare an array in Java. — True
6. Variables declared within a method are called instance variables. — False
7. In Java, you can access instance variables through static methods directly. — False
8. Every object in Java has its own unique set of instance variables. — True
9. The static keyword in Java means that a particular member belongs to a type itself, rather than to instances of that type. — True
10. Variable scope in Java is determined at runtime. — False

KG Coding

Some Other One shot Video Links:

- Complete HTML
- Complete CSS
- Complete JavaScript
- Complete React and Redux
- One shot University Exam Series

http://www.kgcoding.in/

Our YouTube Channels

KG Coding Android App

KG Coding

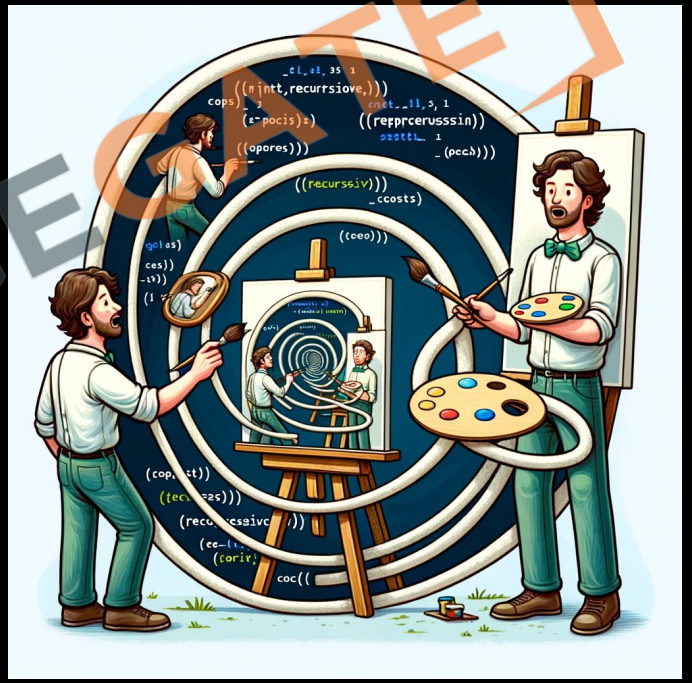
Knowledge GATE

KG Placement Prep

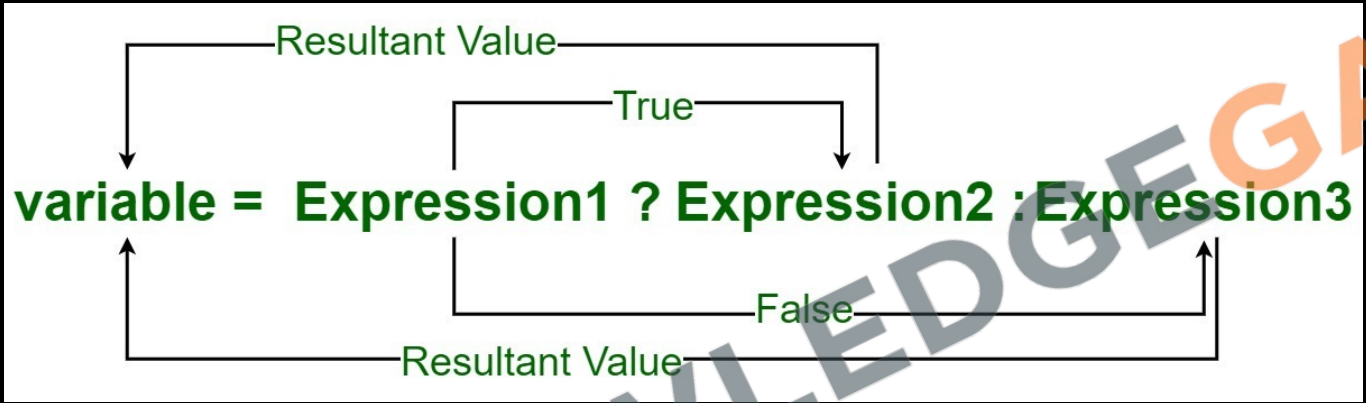
Sanchit Socket

7. Control Statements, Math & String

1. Ternary operator
2. Switch
3. Loops (Do-while, For, For each)
4. Using break & continue
5. Recursion
6. Random Numbers & Math class
7. Don't Learn Syntax
8. toString Method
9. String class
10. StringBuffer vs StringBuilder
11. Final keyword



7.1 Ternary operator



1. Syntax: condition ? expression1 : expression2
2. Condition: Boolean expression, evaluates to true or false.
3. Expressions: Both expressions must return compatible types.
4. Use Case: Suitable for simple conditional assignments.
5. Readability: Good for simple conditions, but can reduce clarity if overused.

7.2 Switch

```
switch (day) {
    case 1: System.out.println("Monday");
            break;
    case 2: System.out.println("Tuesday");
            break;
    case 3: System.out.println("Wednesday");
            break;
    case 4: System.out.println("Thursday");
            break;
    case 5: System.out.println("Friday");
            break;
    case 6: System.out.println("Saturday");
            break;
    case 7: System.out.println("Sunday");
            break;
    default: System.out.println("Invalid day.");
}
```

1. Multiple Cases: Handles multiple values for an expression efficiently.
2. Supported Types: Accepts byte, short, char, int, String, enums, and from Java 14, long, float, double.
3. Case Labels: Each case ends with a colon (:) and is followed by code.
4. Break Statement: Typically used to prevent fall-through between cases.
5. Default Case: Executes if no case matches; optional and doesn't require break.
6. Type Safety: Case label types must match the switch expression's type.

7.2 Switch

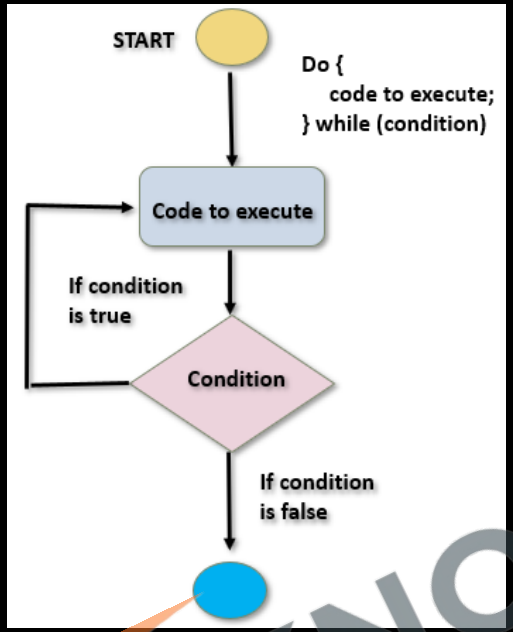
```
String output = switch (day) {
        case 1 -> "Monday";
        case 2 -> "Tuesday";
        case 3 -> "Wednesday";
        case 4 -> "Thursday";
        case 5 -> "Friday";
        case 6 -> "Saturday";
        case 7 -> "Sunday";
        default -> "Invalid";
};
System.out.println(output);
```

1. Enhanced Switch: Java 12 introduced enhancements like yield and multiple constants per case.
2. Switch Expression: From Java 14, switch can return a value using yield.

CHALLENGE

53. Create a program to find the minimum of two numbers.
54. Create a program to find if the given number is even or odd.
55. Create a program to calculate the absolute value of a given integer.
56. Create a program to Based on a student's score, categorize as "High", "Moderate", or "Low" using the ternary operator (e.g., High for scores > 80, Moderate for 50-80, Low for < 50).
57. Create a program to print the month of the year based on a number (1-12) input by the user.
58. Create a program to create a simple calculator that uses a switch statement to perform basic arithmetic operations like addition, subtraction, multiplication, and division.

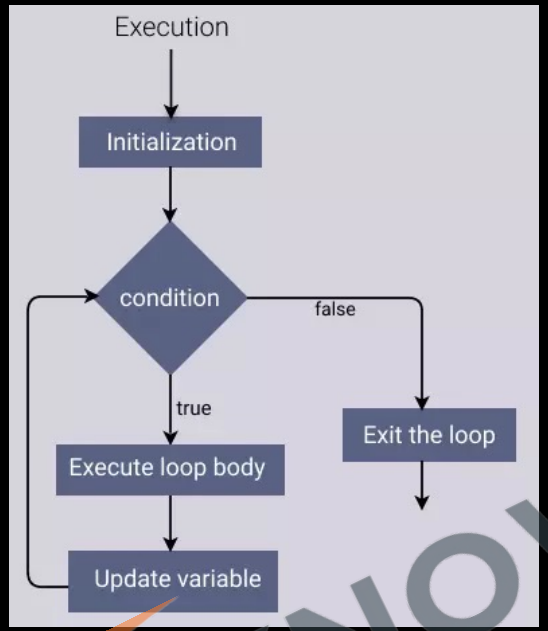
7.3 Loops (Do-while)




```do
do {
  // Body of the loop
}
while (condition);
```


1. Executes block first, then checks condition.
2. Guaranteed to run at least one iteration.
3. Unlike while, first iteration is unconditional.
4. Don't forget to update condition to avoid infinite loops.

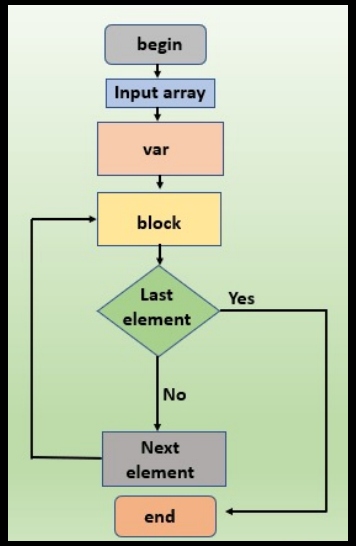
7.3 Loops (For)




```
for (initialisation; condition; update) {
  // Body of the loop
}
```


1. Standard loop for running code multiple times.
2. Generally preferred for counting iterations.

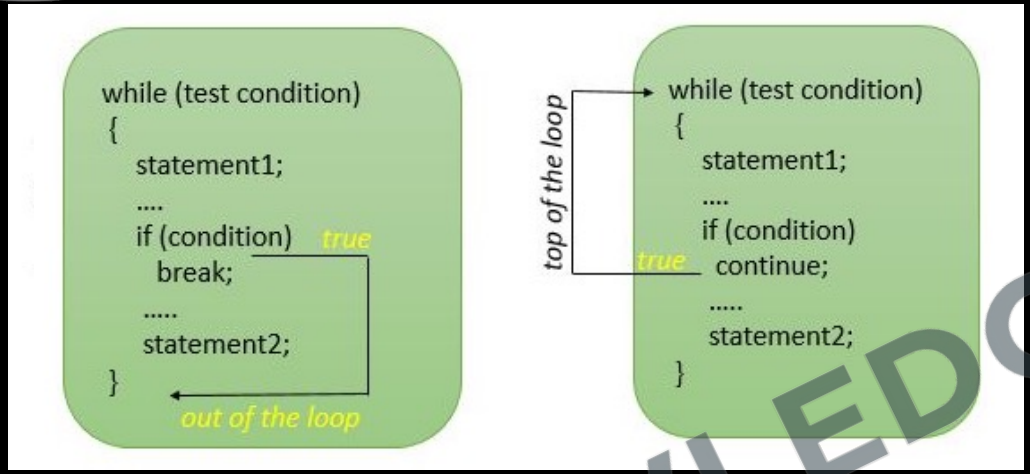
7.3 Loops (For each)




```
String[] names = new String[] {
    "Ram", "Shyam", "Mohan", "Geeta",
    "Sita", "Sohan"
};
for (String name: names) {
    System.out.println(name);
}
```

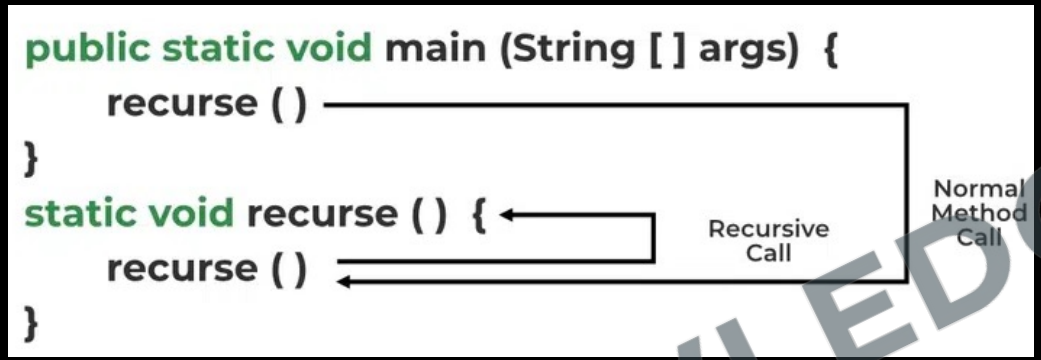

1. A method for array iteration, often preferred for readability.
2. Parameters: One for item, optional second for index.
3. Using return is similar to continue in traditional loops.
4. Not straightforward to break out of a forEach loop.
5. When you need to perform an action on each array element and don't need to break early.

7.4 Using break & continue



1. Break lets you stop a loop early, or break out of a loop
2. Continue is used to skip one iteration or the current iteration
3. In while loop remember to do the increment manually before using continue

7.5 Recursion

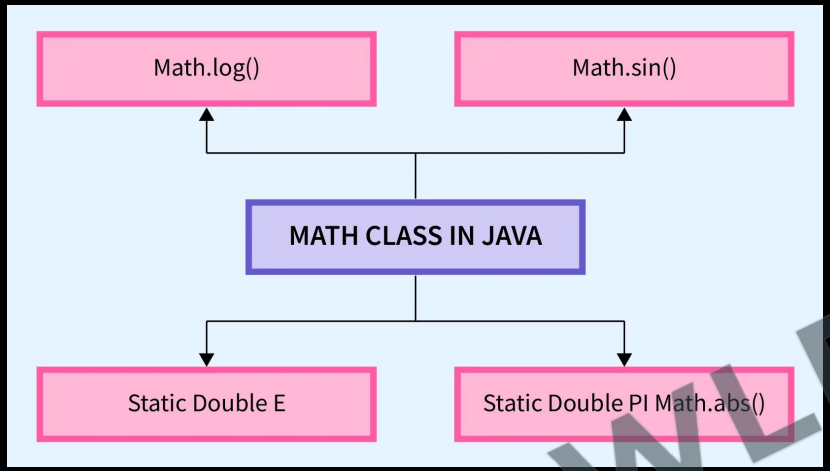


1. Self-Calling Function: Recursion is when a function calls itself.
2. Base Case: Essential to stop recursion and prevent infinite loops.
3. Recursive Case: The part where the function makes a recursive call.
4. Stack Overflow Risk: Excessive recursion can cause stack overflow errors.
5. Problem Solving: Ideal for problems divisible into similar, smaller problems.

CHALLENGE

59. Create a program using do-while to find password checker until a valid password is entered.
60. Create a program using do-while to implement a number guessing game.
61. Create a program using for loop multiplication table for a number.
62. Create a program using for to display if a number is prime or not.
63. Create a program using for-each to find the maximum value in an integer array.
64. Create a program using for-each to the occurrences of a specific element in an array.
65. Create a program using break to read inputs from the user in a loop and break the loop if a specific keyword (like "exit") is entered.
66. Create a program using continue to sum all positive numbers entered by the user; skip any negative numbers.
67. Create a program using continue to print only even numbers using continue for odd numbers.
68. Create a program using recursion to display the Fibonacci series upto a certain number.
69. Create a program using recursion to check if a string is a palindrome using recursion.

7.6 Random Numbers & Math class



Key Methods:

1. abs(): Absolute value.
2. ceil(): Rounds up.
3. floor(): Rounds down.
4. round(): Rounds to nearest integer.
5. max(), min(): Maximum and minimum of two numbers.
6. pow(): Power calculation.
7. sqrt(): Square root.
8. random(): Random number generation.
9. exp(), log(): Exponential and logarithmic functions.
10. Trigonometric functions: sin(), cos(), tan().

1. Static Class: Math methods are static and accessed directly.
2. Constants: Includes PI and E for π and the base of natural logarithms.

7.7 Don't Learn Syntax

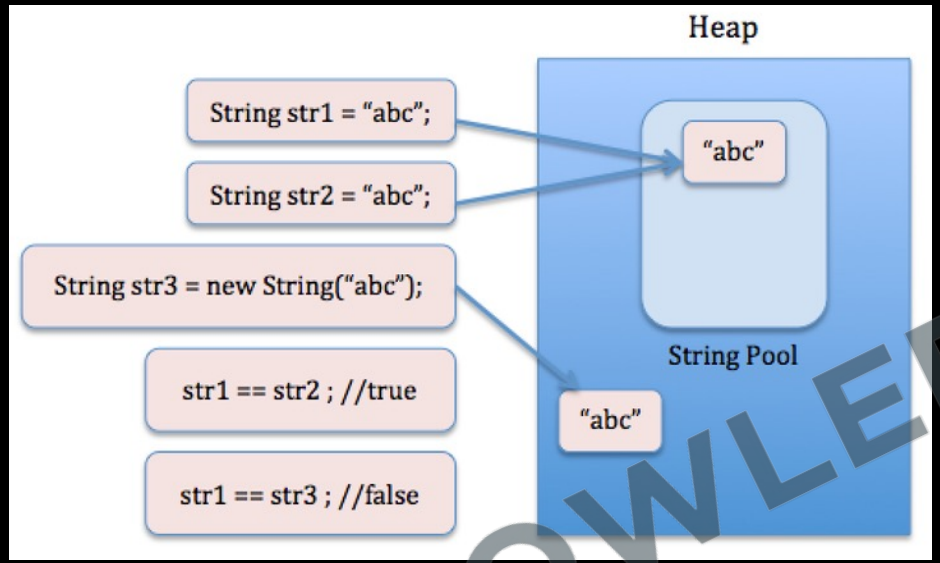
1. Google: Quick answers to coding problems.
2. Oracle: In-depth guides and documentation. https://docs.oracle.com/
3. ChatGPT: Real-time assistance for coding queries.
4. Focus: Understand concepts, not just syntax.

7.8 toString Method

```java
@Override
public String toString() {
    return "Car{" +
            "noOfWheels=" + noOfWheels +
            ", color='" + color + '\'' +
            ", maxSpeed=" + maxSpeed +
            ", currentFuelInLiters=" + currentFuelInLiters +
            ", noOfSeats=" + noOfSeats +
            '}';
}
```


1. Function: toString() provides a string representation of an object.
2. Inheritance: It's inherited from the Object class.
3. Default Format: By default, returns class name, "@", and hashcode.
4. Overriding: Commonly overridden in custom classes for meaningful output.
5. Implicit Call: Automatically called in string concatenation.

7.9 String class



1. Immutability: Once created, a String object's value cannot be changed. Modifications create new String objects.
2. String Pool: Java maintains a pool of strings for efficiency. When a new string is created, it's checked against the pool for a match to reuse.
3. Comparing: equals() method for value comparison, == operator checks reference equality.

7.9 String class

```
"Car{" +
 "noOfWheels=" + noOfWheels +
 ", color='" + color + '\'' +
 ", maxSpeed=" + maxSpeed +
 ", currentFuelInLiters=" + currentFuelInLiters +
 ", noOfSeats=" + noOfSeats +
 '}';
```

1. Concatenation: Strings can be concatenated using the + operator, but each concatenation creates a new String.
2. Methods: Provides methods like length(), substring(), equals(), compareTo(), indexOf(), for various operations.
3. Memory: Being immutable, strings can use more memory when frequently modified.

7.9 String class

Printf specifier	Data type
%s	String of text
%f	floating point value (float or double)
%e	Exponential, scientific notation of a float or double
%b	boolean true or false value
%c	Single character char
%d	Base 10 integer, such as a Java int, long, short or byte
%o	Octal number
%x	Hexadecimal number
%%	Percentage sign
%n	New line, aka carriage-return
%tY	Year to four digits
%tT	Time in format of HH:MM:SS (ie 21:46:30)

printf flag	Purpose
–	Aligns the formatted *printf* output to the left
+	The output includes a negative or positive sign
(	Places negative numbers in parenthesis
0	The formatted *printf* output is zero padded
,	The formatted output includes grouping separators
<space>	A blank space adds a minus sign for negative numbers and a leading space when positive

```
% [flags] [width] [.precision] specifier-character
```

7.9 String class

Pattern	Data	*Printf* Output
'%s'	Java	'Java'
'%15s'	Java	' Java'
'%-15s '	Java	'Java '
'%-15S '	Java	'JAVA '

7.9 String class

Pattern	Data	*Printf* output
'%d'	123,457,890	'123457890'
'%,15d'	123,457,890	' 123,457,890'
'%+,15d'	123457890	' +123,457,890'
'%-+,15d'	123457890	'+123,457,890 '
'%0,15d'	123457890	'0000123,457,890'

7.10 StringBuffer vs StringBuilder

```
StringBuilder sentence = new StringBuilder("This is a sentence.");
sentence.append("Added word.");
System.out.println(sentence.toString()); //This is a sentence.Added word.
```

Parameter	String	StringBuilder	StringBuffer
mutability	immutable	mutable	mutable
Storage	String constant pool	heap	heap
Thread safety	Not used in the threaded environment as it is immutable	Not thread-safe so it is used in a single-threaded environment	Thread-safe so it is used in a multi-threaded environment
Speed	Comparably slowest	Comparably fastest	Faster than String but Slower than StringBuilder

7.11 Final keyword

```java
public class Main {
    public final String name = "John";

    public void setName(String name) {
        this.name = name; // cannot reassign final variables
    }
}
```

1. Variable: When applied to a variable, it becomes a constant, meaning its value cannot be changed once initialized.
2. Efficiency: Using final can lead to performance optimization, as the compiler can make certain assumptions about final elements.
3. Null Safety: A final variable must be initialized before the constructor completes, reducing null pointer errors.
4. Immutable Objects: Helps in creating immutable objects in combination with private fields and no setter methods.

CHALLENGE

70. Define a Student class with fields like name and age, and use toString to print student details.
71. Concatenate and Convert: Take two strings, concatenate them, and convert the result to uppercase.
72. Calculate the area and circumference of a circle for a given radius using Math.PI
73. Simulate a dice roll using Math.random() and display the outcome (1 to 6).
74. Create a number guessing game where the program selects a random number, and the user has to guess it.
75. Take an array of words and concatenate them into a single string using StringBuilder.
76. Create an object with final fields and a constructor to initialize them.

Revision

1. Ternary operator
2. Switch
3. Loops (Do-while, For, For each)
4. Using break & continue
5. Recursion
6. Random Numbers & Math class
7. Don't Learn Syntax
8. toString Method
9. String class
10. StringBuffer vs StringBuilder
11. Final keyword

Practice Exercise

Control Statements, Math & String

Answer in True/False:

1. The ternary operator in Java can replace certain types of if-else statements.
2. A switch statement in Java can only test for equality with constants.
3. The do-while loop will execute at least once even if the condition is false.
4. A for-each loop in Java is used to iterate over arrays.
5. The break statement skips the current iteration.
6. The continue statement exits the loop immediately, regardless of the condition.
7. In Java, recursion is a process where a method can call itself directly.
8. Random numbers generated by Math.random() are inclusive of 0 and 1.
9. The String class is mutable, meaning it can be changed after it's created.
10. StringBuilder is faster than StringBuffer since it is not synchronized.
11. A final variable in Java can be assigned a value once and only once.

Practice Exercise

Control Statements, Math & String

Answer in True/False:

1. The ternary operator in Java can replace certain types of if-else statements. — True
2. A switch statement in Java can only test for equality with constants. — True
3. The do-while loop will execute at least once even if the condition is false. — True
4. A for-each loop in Java is used to iterate over arrays. — True
5. The break statement skips the current iteration. — False
6. The continue statement exits the loop immediately, regardless of the condition. — False
7. In Java, recursion is a process where a method can call itself directly. — True
8. Random numbers generated by Math.random() are inclusive of 0 and 1. — False
9. The String class is mutable, meaning it can be changed after it's created. — False
10. StringBuilder is faster than StringBuffer since it is not synchronized. — True
11. A final variable in Java can be assigned a value once and only once. — True

KG Coding

Some Other One shot Video Links:

- Complete HTML
- Complete CSS
- Complete JavaScript
- Complete React and Redux
- One shot University Exam Series

http://www.kgcoding.in/

Our YouTube Channels

KG Coding Android App

KG Coding

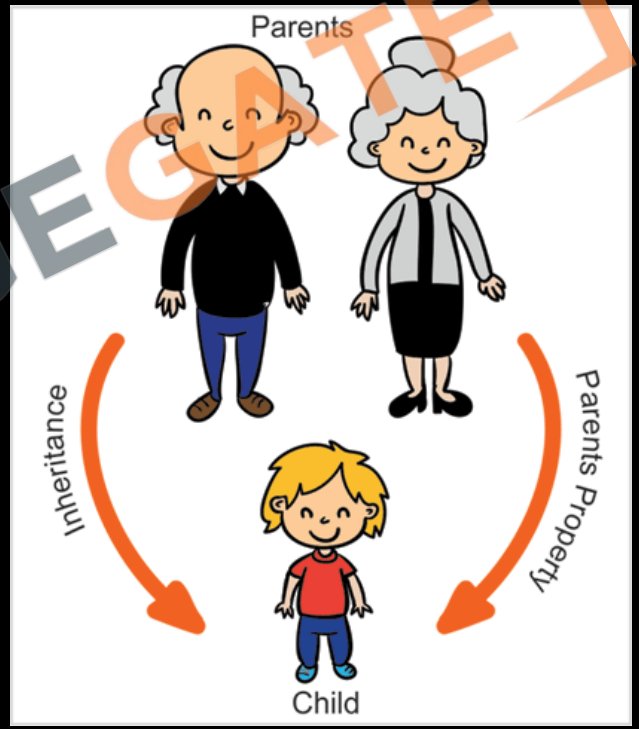
Knowledge GATE

KG Placement Prep

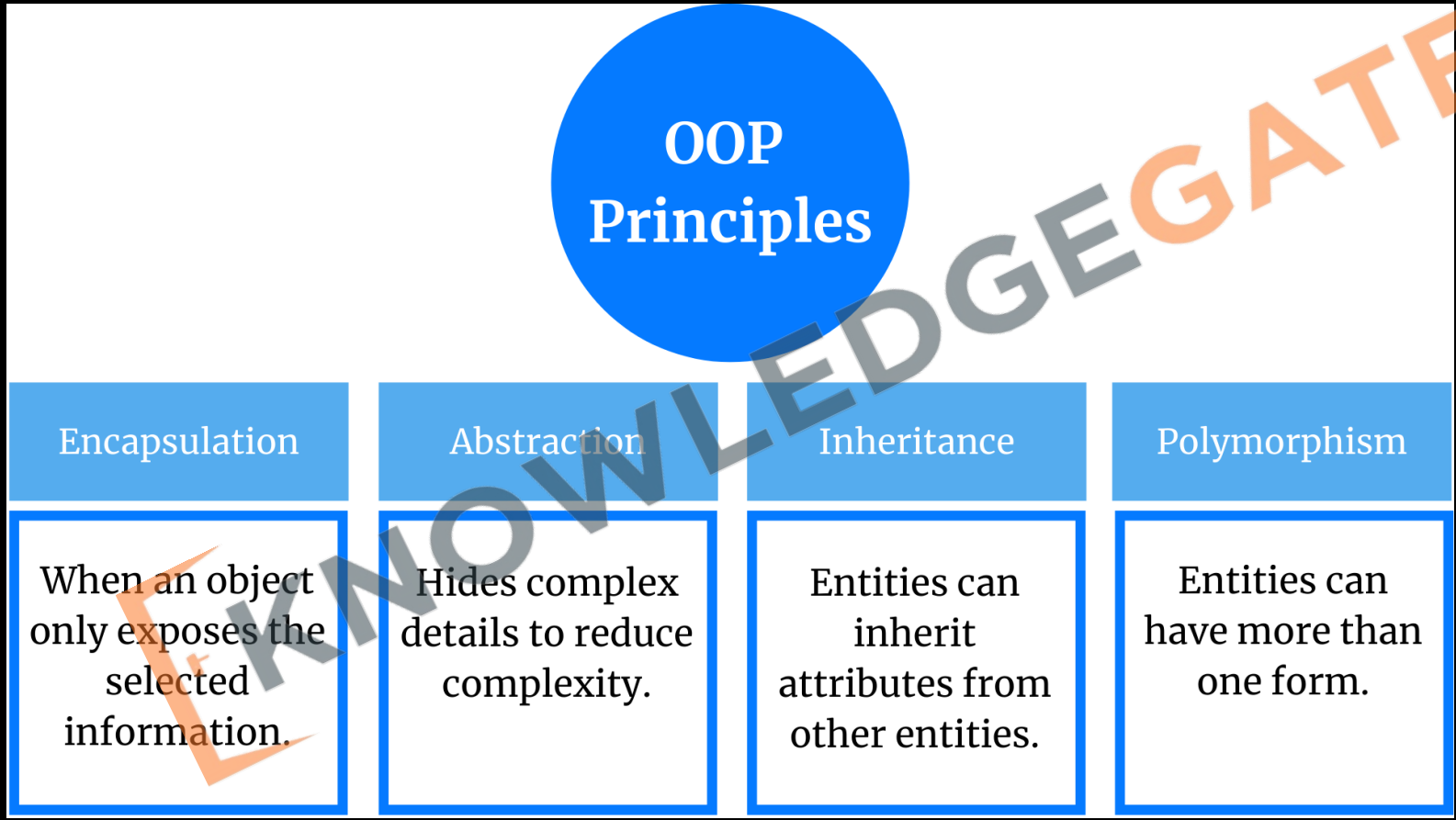
Sanchit Socket

8 Encapsulation & Inheritance

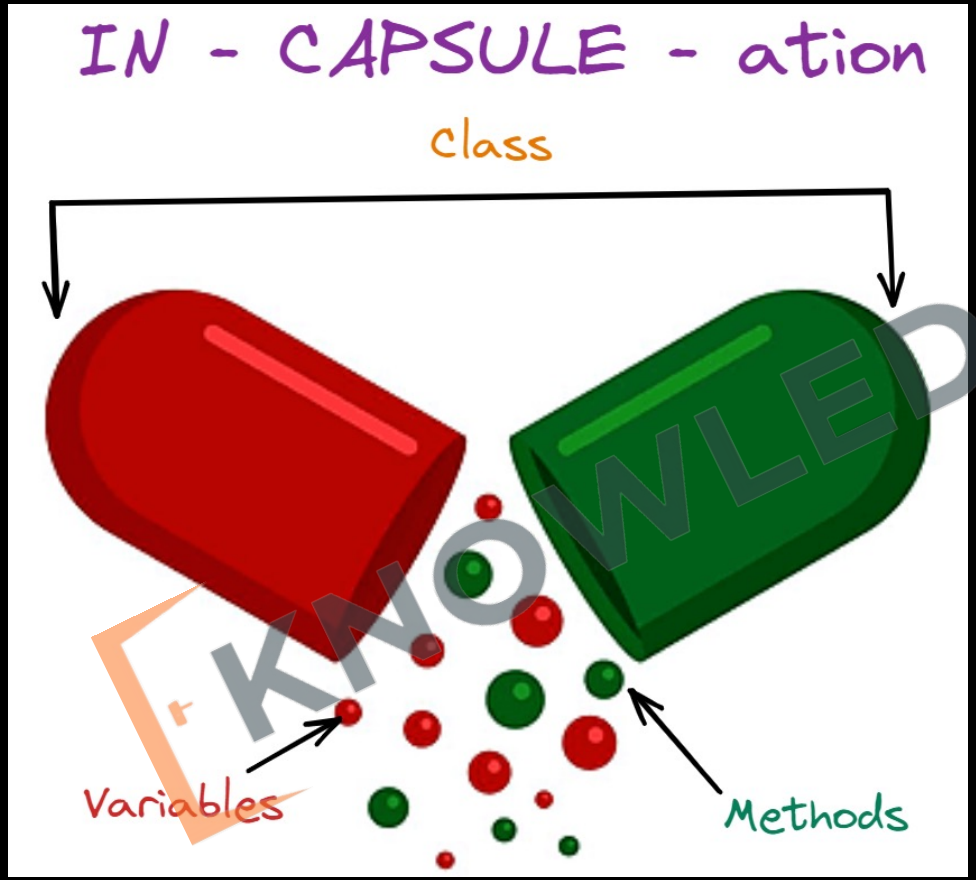
1. Intro to OOPs Principle
2. What is Encapsulation
3. Import & Packages
4. Access Modifiers
5. Getter and Setter
6. What is Inheritance
7. Types of Inheritance
8. Object class
9. Equals and Hash Code
10. Nested and Inner Classes



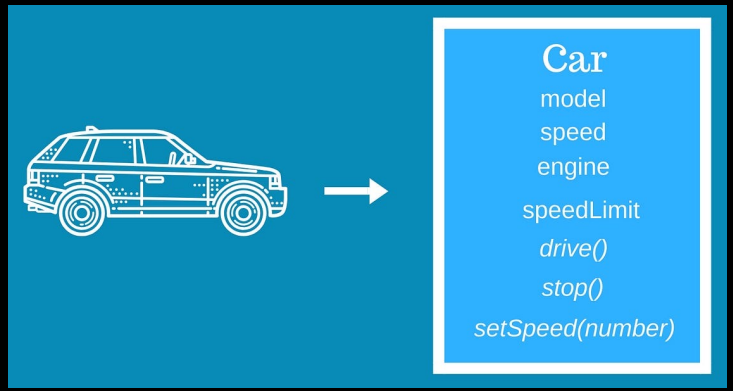
8.1 Intro to OOPs Principle



8.2 What is Encapsulation



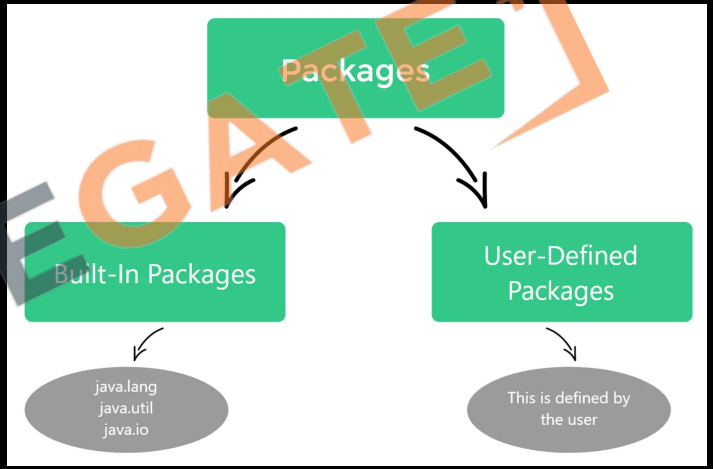
8.2 What is Encapsulation



1. Data Hiding: Encapsulation hides internal data, allowing access only through methods.
2. Access Modifiers: Uses private, public, protected to control access to class members.
3. Getter/Setter: Provides public methods for controlled property access.
4. Maintains Integrity: Protects object state from external interference.
5. Enhances Modularity: Keeps classes separate and reduces coupling.

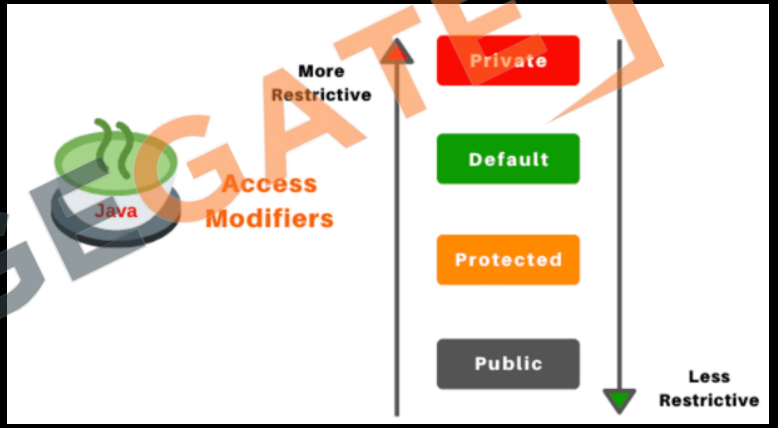
8.3 Import & Packages

1. Package Definition: A package in Java is a namespace that organizes classes and interfaces, preventing naming conflicts.
2. Package Declaration: Packages are declared at the beginning of a Java source file using the package keyword followed by the package name.
3. Import Statement: An import statement in Java is used to bring in classes or interfaces from other packages into the current file, making them accessible without using a fully qualified name.
4. Types of Import:
   - Single-Type Import: Imports a single class or interface from a package (e.g., import java.util.List;).
   - On-Demand Import: Imports all classes and interfaces from a package (e.g., import java.util.*;).
5. Avoiding Collisions: Packages help in avoiding name collisions by categorizing similar classes together.
6. Built-in Packages: Java comes with built-in packages like java.lang (automatically imported), java.util, java.io, etc.



8.4 Access Modifiers

1. Types: The four access modifiers are public, protected, default (no modifier), and private.
2. Public: The public modifier allows access from any other class. For classes, it means they can be accessed from any other package.
3. Protected: The protected modifier allows access within the same package and subclasses.
4. Default: If no access modifier is specified, it's the default, allowing access only within the same package.
5. Private: The private modifier restricts access to the defining class only.
6. Class-Level Access: Only public and default (no modifier) access modifiers are applicable for top-level classes.
7. Member-Level Access: Methods, constructors, and variables can use all four access modifiers to control visibility.



8.4 Access Modifiers

Access Specifiers in Java

		public	private	protected	default
Same Package	Class	YES	YES	YES	YES
	Sub class	YES	NO	YES	YES
	Non sub class	YES	NO	YES	YES
Different Package	Sub class	YES	NO	YES	NO
	Non sub class	YES	NO	NO	NO

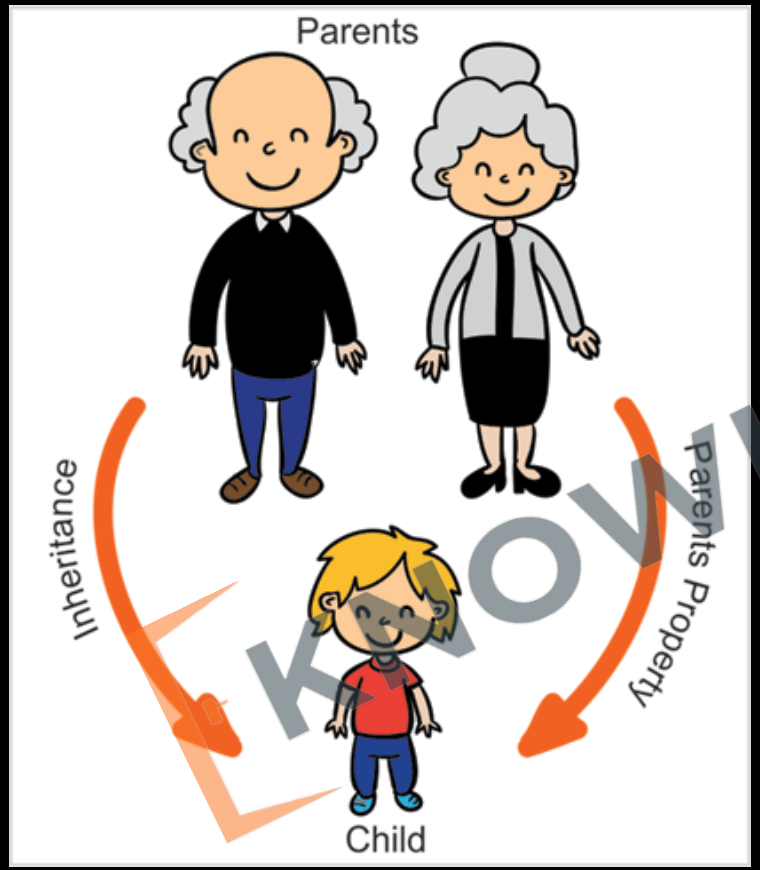
8.5 Getter and Setter

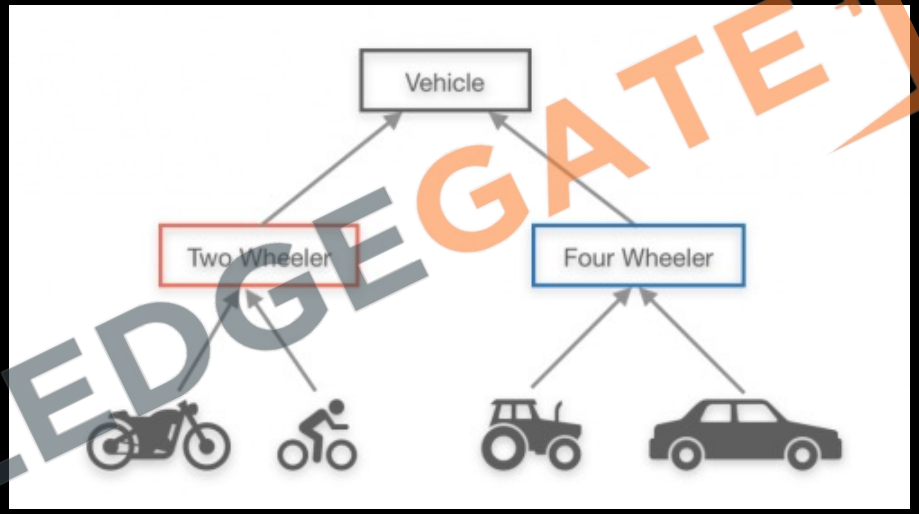
1. Getters: Retrieve private field values, typically named get<FieldName>.
2. Setters: Set or update private field values, usually named set<FieldName>.
3. Control and Validation: Offer controlled access and allow for validation logic.
4. Encapsulation: Facilitate read-only or write-only access to class fields.
5. Flexibility: Allow for internal changes without affecting external interfaces.

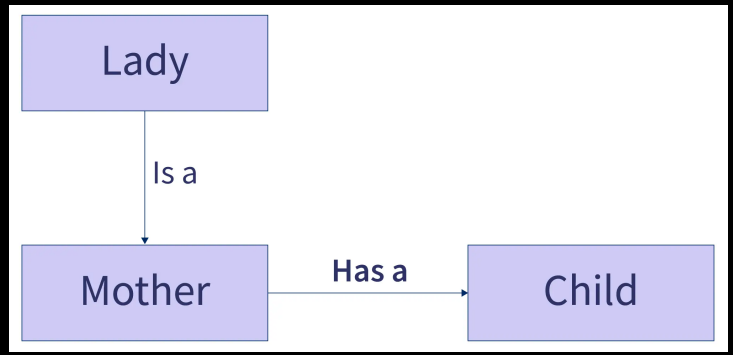
CHALLENGE

77. Create a simple application with at least two packages: com.example.geometry and com.example.utils. In the geometry package, define classes like Circle and Rectangle. In the utils package, create a Calculator class that can compute areas of these shapes.
78. Define a BankAccount class with private attributes like accountNumber, accountHolderName, and balance. Provide public methods to deposit and withdraw money, ensuring that these methods don't allow illegal operations like withdrawing more money than the current balance.
79. Define a class Employee with private attributes (like name, age, and salary), public methods to get and set these attributes, and a package-private method to displayEmployeeDetails. Create another class in the same package to test access to the displayEmployeeDetails method.

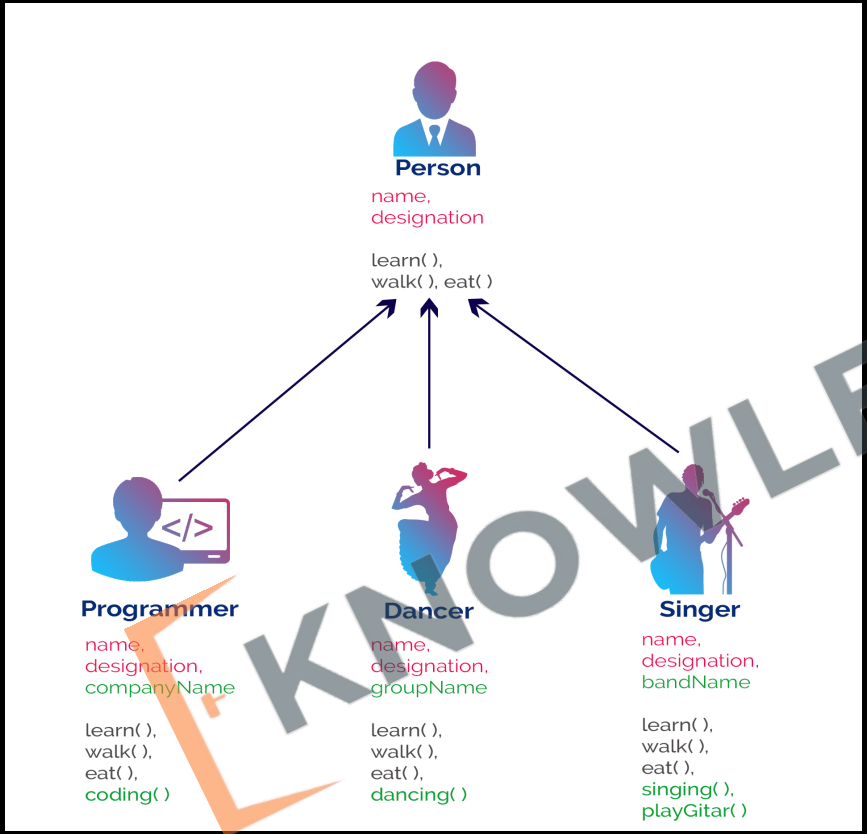
8.6 What is Inheritance





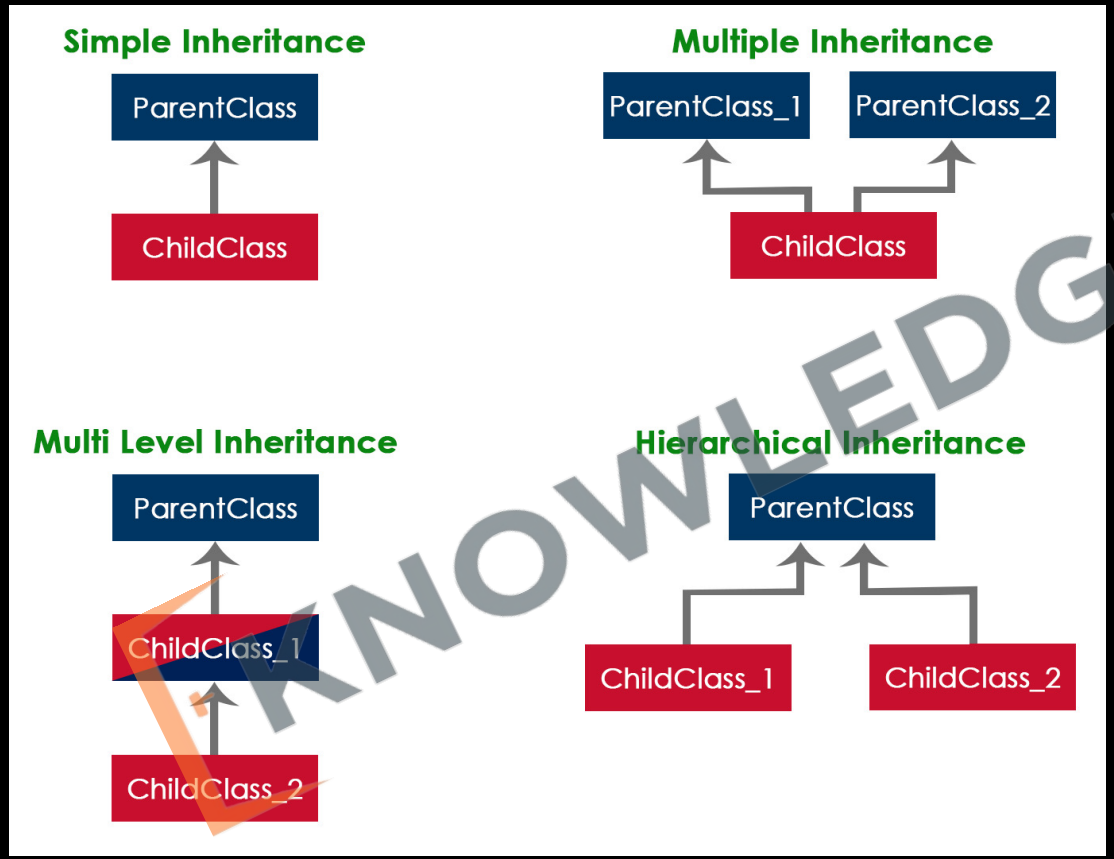


8.6 What is Inheritance

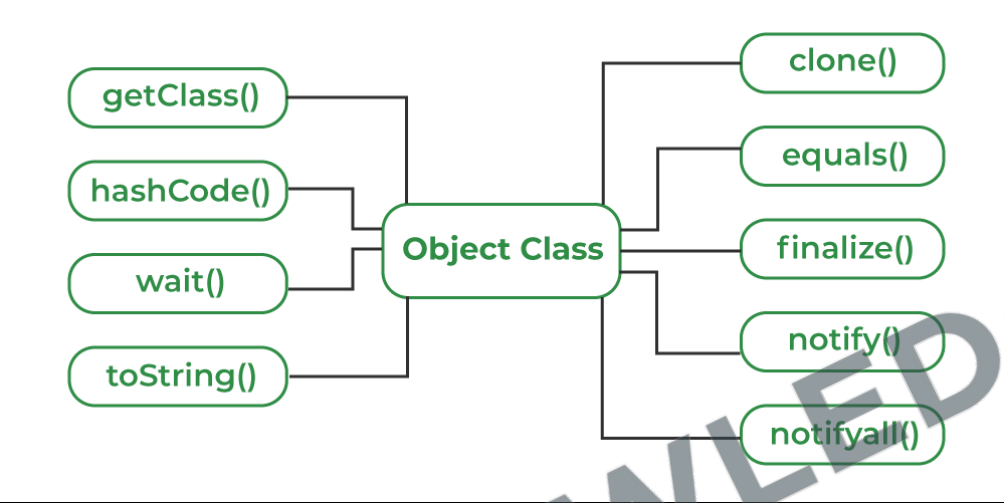


1. Inheritance allows a new class (subclass) to inherit features from an existing class (superclass).
2. Code Reusability: It enables subclasses to use methods and variables of the superclass, reducing code duplication.
3. Access Control: The protected access modifier is often used in inheritance to allow subclass access to superclass members.

8.7 Types of Inheritance

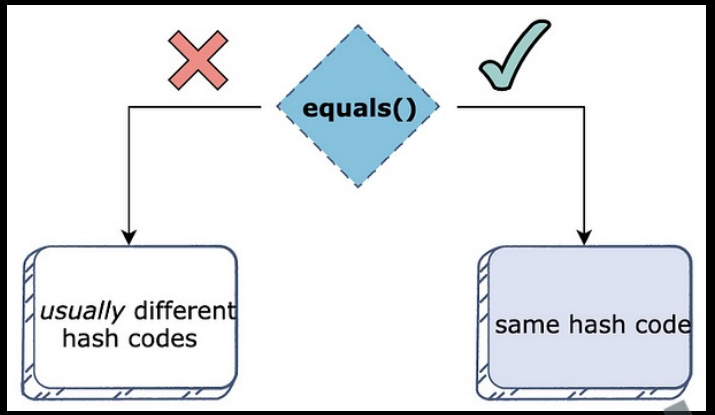


8.8 Object class



1. Root Class: The Object class is the parent class of all classes in Java, forming the top of the class hierarchy.
2. Default Methods: It provides fundamental methods like equals(), hashCode(), toString(), and getClass() that can be overridden by subclasses.
3. String Representation: The toString() method returns a string representation of the object, often overridden for more informative output.

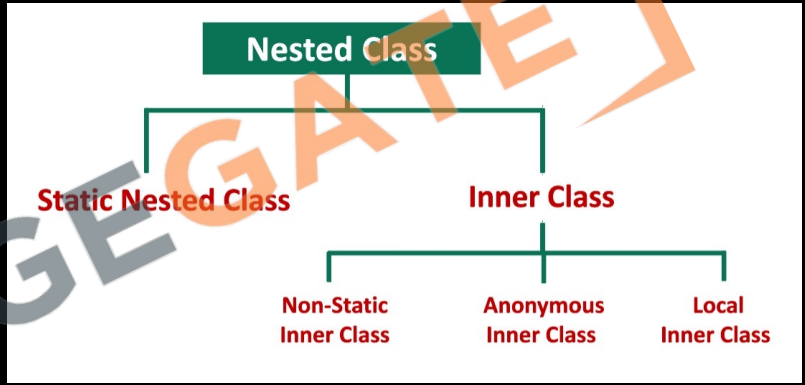
8.9 Equals and HashCode



1. equals() Method: Used for logical equality checks between objects. By default, it compares object references, but it's commonly overridden to compare object states.
2. hashCode() Method: Generates an integer hash code representing an object. It's crucial for the performance of hash-based collections like HashMap.
3. Equals-Hashcode Contract: If two objects are equal according to equals(), they must have the same hash code. However, two objects with the same hash code aren't necessarily equal.
4. Overriding Both: If equals() is overridden, hashCode() should also be overridden to maintain consistency between these methods.

8.10 Nested and Inner Classes

1. Nested Classes: Classes defined within another class, divided into static (static nested classes) and non-static (inner classes).
2. Static Nested Classes: Act as static members of the outer class; can access outer class's static members but not its non-static members.
3. Inner Classes: Associated with an instance of the outer class; can access all members of the outer class, including private ones.
4. Local and Anonymous Inner Classes: Local inner classes are defined within a block (like a method) and are not visible outside it. Anonymous inner classes are nameless and used for single-use implementations.
5. Use Cases: Useful for logically grouping classes, improving encapsulation, and enhancing code readability.



CHALLENGE

80. Start with a base class LibraryItem that includes common attributes like itemID, title, and author, and methods like checkout() and returnItem(). Create subclasses such as Book, Magazine, and DVD, each inheriting from LibraryItem. Add unique attributes to each subclass, like ISBN for Book, issueNumber for Magazine, and duration for DVD.

81. Create a class Person with attributes name and age. Override equals() to compare Person objects based on their attributes. Override hashCode() consistent with the definition of equals().

82. Create a class ArrayOperations with a static nested class Statistics. Statistics could have methods like mean(), median(), which operate on an array.

Revision

1. Intro to OOPs Principle
2. What is Encapsulation
3. Import & Packages
4. Access Modifiers
5. Getter and Setter
6. What is Inheritance
7. Types of Inheritance
8. Object class
9. Equals and Hash Code
10. Nested and Inner Classes

Practice Exercise

Encapsulation & Inheritance

Answer in True/False

1. Encapsulation is the OOP principle that ensures an object's data can be changed by any method, without restrictions.
2. The import statement in Java can be used to bring a single class or an entire package into visibility.
3. The private access modifier means that the member is only accessible within its own class.
4. Setters are used to retrieve the value of a private variable from outside the class.
5. Inheritance in OOP can be used to create a general class that defines traits to be inherited by more specific subclasses.
6. The Object class in Java has protected access by default.
7. If two objects are equal according to their equals() method, then their hashCode() methods must return different integers.
8. Inner classes are defined within the scope of a method and cannot exist independently of the method.
9. A static nested class can access the instance variables of its enclosing outer class.
10. Overriding the equals() method requires also overriding the hashCode() method to maintain consistency.

Practice Exercise

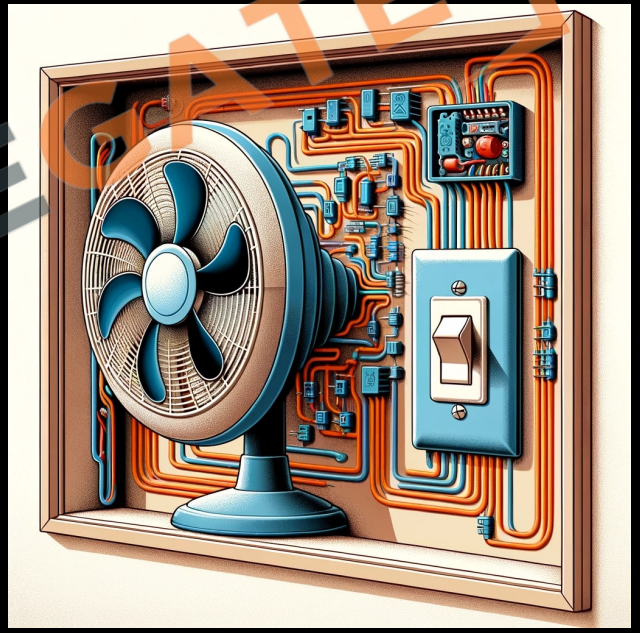
Encapsulation & Inheritance

Answer in True/False

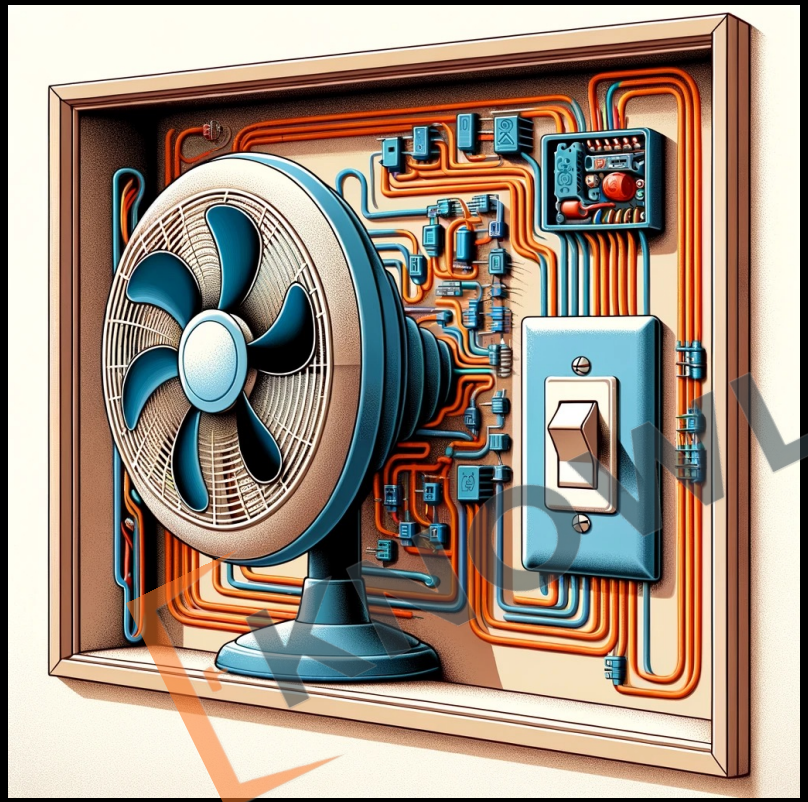
1. Encapsulation is the OOP principle that ensures an object's data can be changed by any method, without restrictions. — False
2. The import statement in Java can be used to bring a single class or an entire package into visibility. — True
3. The private access modifier means that the member is only accessible within its own class. — True
4. Setters are used to retrieve the value of a private variable from outside the class. — False
5. Inheritance in OOP can be used to create a general class that defines traits to be inherited by more specific subclasses. — True
6. The Object class in Java has protected access by default. — False
7. If two objects are equal according to their equals() method, then their hashCode() methods must return different integers. — False
8. Inner classes are defined within the scope of a method and cannot exist independently of the method. — False
9. A static nested class can access the instance variables of its enclosing outer class. — False
10. Overriding the equals() method requires also overriding the hashCode() method to maintain consistency. — True

9 Abstraction and Polymorphism

1. What is Abstraction
2. Abstract Keyword
3. Interfaces
4. What is Polymorphism
5. References and Objects
6. Method / Constructor Overloading
7. Super Keyword
8. Method / Constructor Overriding
9. Final keyword revisited
10. Pass by Value vs Pass by reference.

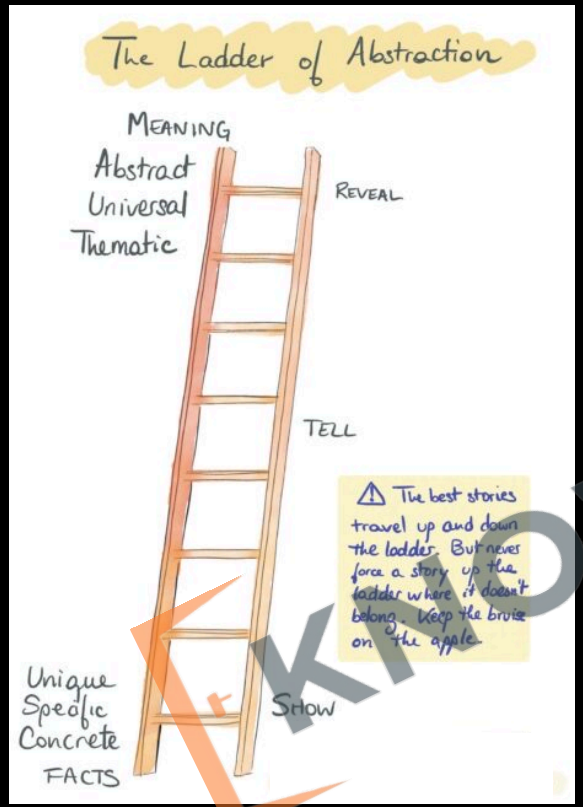


9.1 What is Abstraction



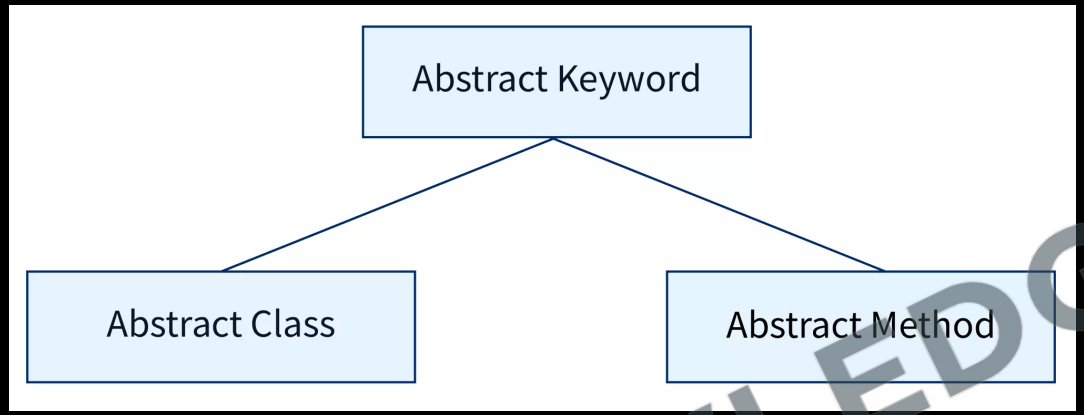
1. Core Principle: Abstraction hides complex implementation details, focusing only on essential features.
2. Focus on Functionality: Emphasizes what an object does, not how it does it, through clear interfaces.

9.1 What is Abstraction



1. Simplifies Complexity: It reduces complexity by showing only relevant information in class design.
2. Real-World Modelling: Abstraction allows creating objects that represent real-life entities with key attributes and behaviours.

9.2 Abstract Keyword



1. Abstract Class: Used to declare non-instantiable abstract classes that serve as base classes.
2. Abstract Method: Defines methods without implementations, requiring subclasses to provide specific functionality.
3. Mandatory Implementation: Subclasses must implement all abstract methods of an abstract class.
4. Design Flexibility: Allows for flexible class design by defining a contract for subclasses.