Object Oriented Programming with Java (OOPJ)

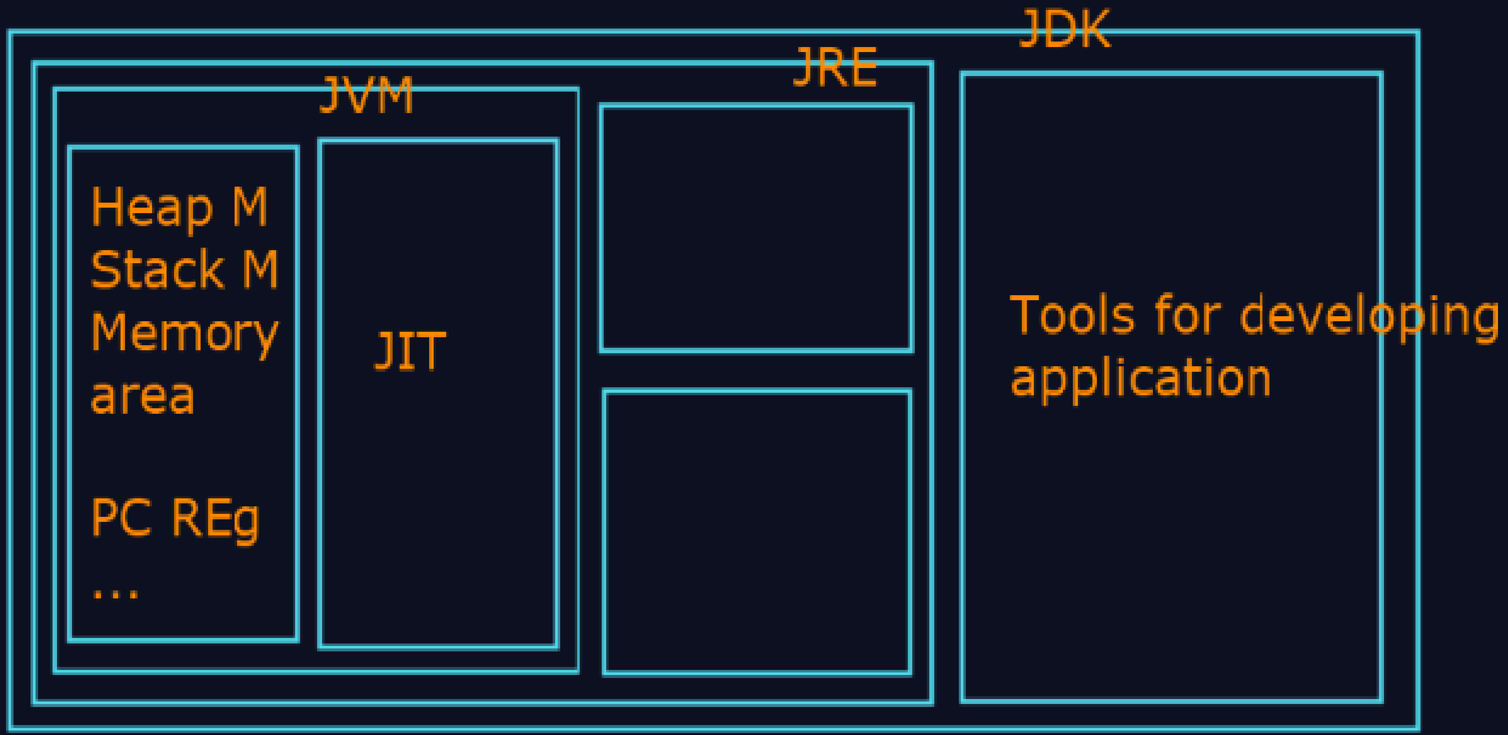
Session 2: Programming concepts

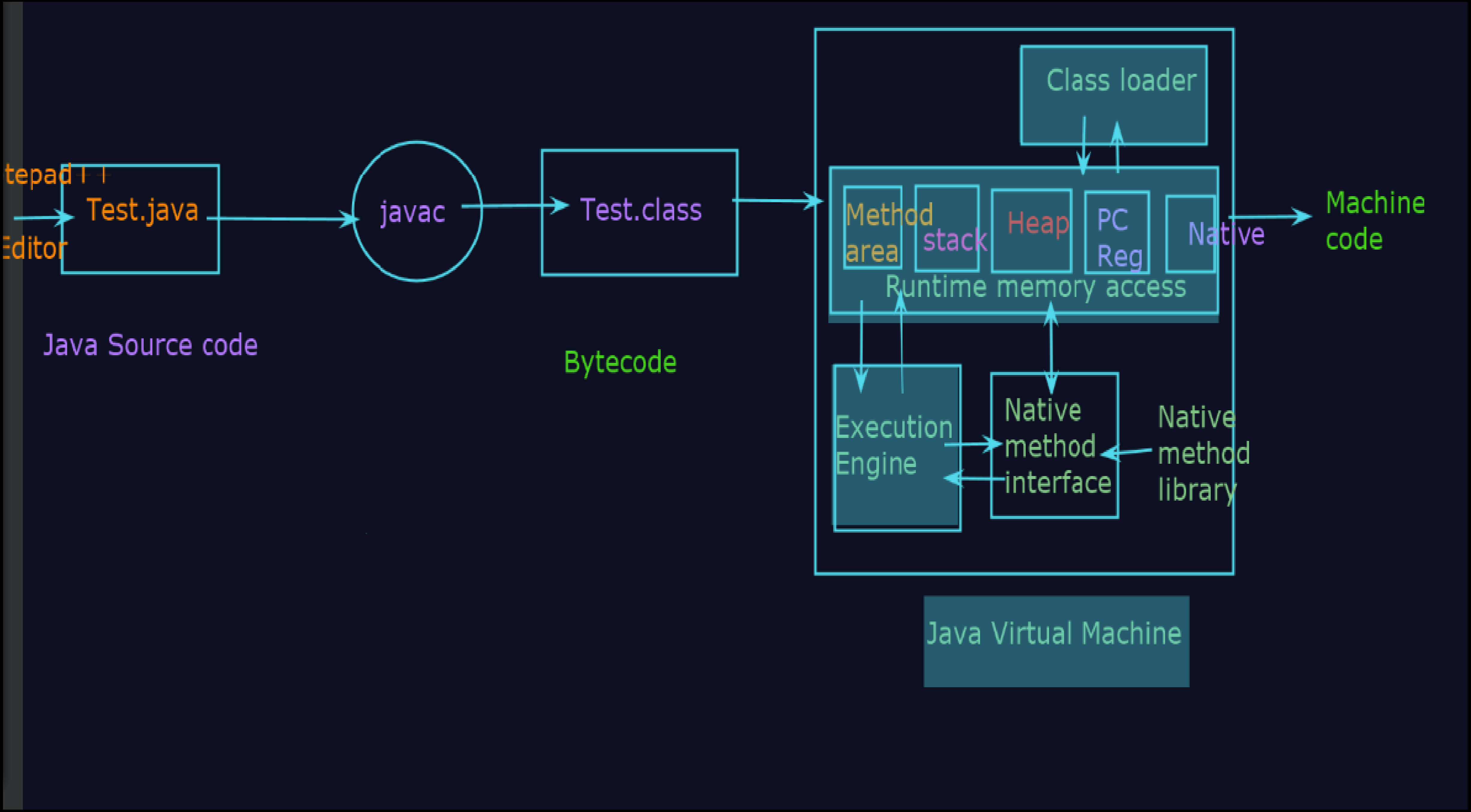
Kiran Waghmare

Date: 26/08/2025
Day 2 : OOPJ

Topics:

- Java Tokens: keywords, identifiers, literals, operators
- Declaring variables and methods
- Data type compatibility
- Programs




tepad
Editor
Test.java
Java Source code
javac
Test.class
Bytecode
Class loader
Method area
stack
Heap
PC Reg
Native
Runtime memory access
Execution Engine
Native method interface
Native method library
Machine code
Java Virtual Machine

Java Tokens

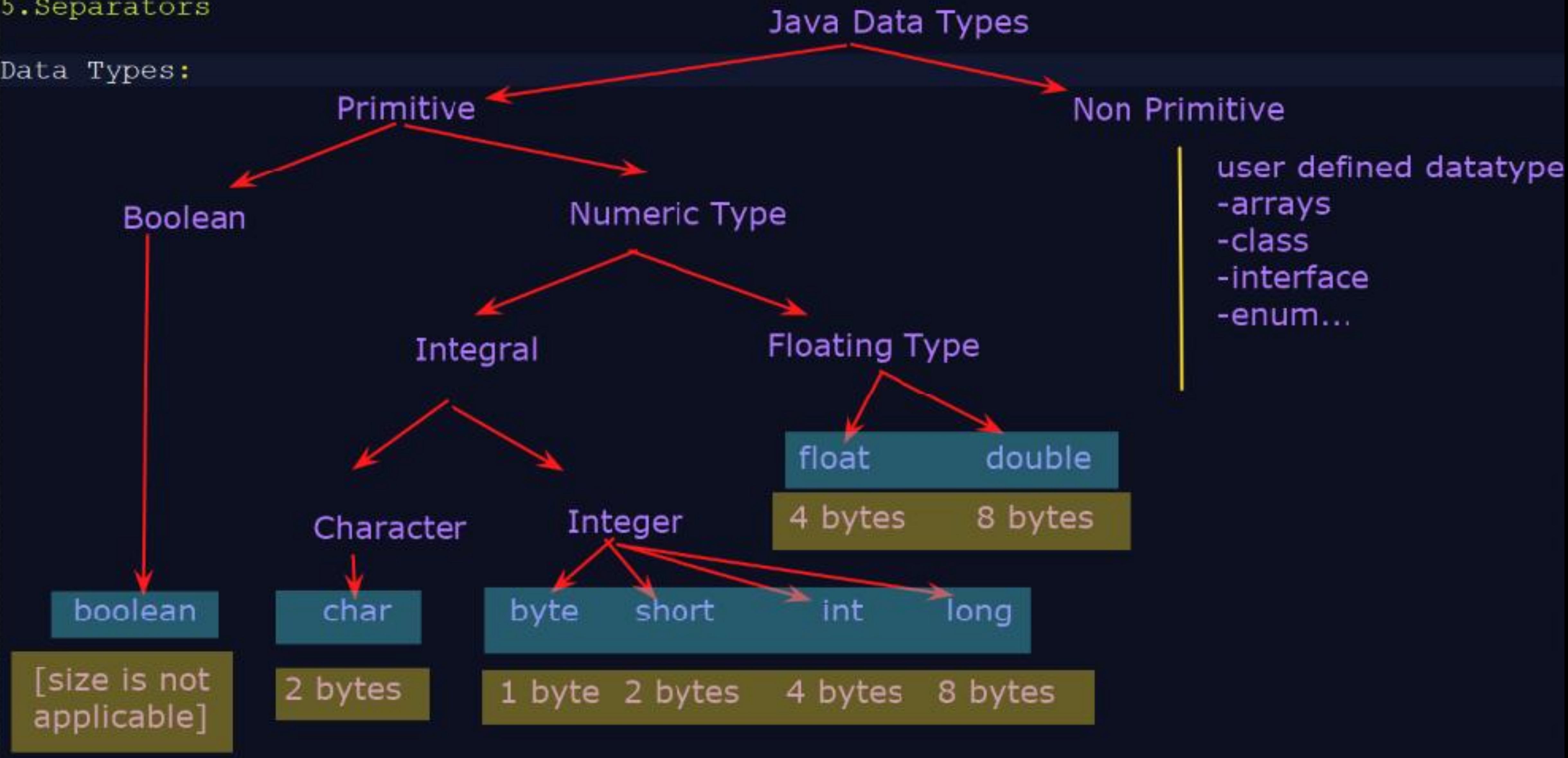
- Tokens - The smallest individual unit of program are known as Tokens.

- Java Program – It is a collection of Tokens , comments and white spaces. It contains 5 types of tokens:

1. **Reserved words – keywords**
   - 50 keywords
   - Having specific meaning – we cannot use them as names for variables ,class name etc
   - Always lower case letters, case sensitive
   - E.g., abstract, case, short, super etc
2. **Identifiers – a**
   - Programmer designed tokens
   - Used for naming classes, methods, variables, labels, packages, interfaces in a program
   - Rules-
     1. Have alphabets,digits and _ and $
     2. Not begin with digit
     3. Uppercase & lowercase letters are distinct
     4. Can be of any length

1. **Literals –**
   - Sequence of character
   - Represents constant value to be stored in variable
   - 5 – types- Integer, Floating-point, Character, String and Boolean
2. **Operators –**
   - Symbol that takes one / more arguments & operates on them to produce a result.
3. **Separators –**
   - Group of code are divided & arranged
   - i.e., ( ), { }, [ ], ;, , ,& .

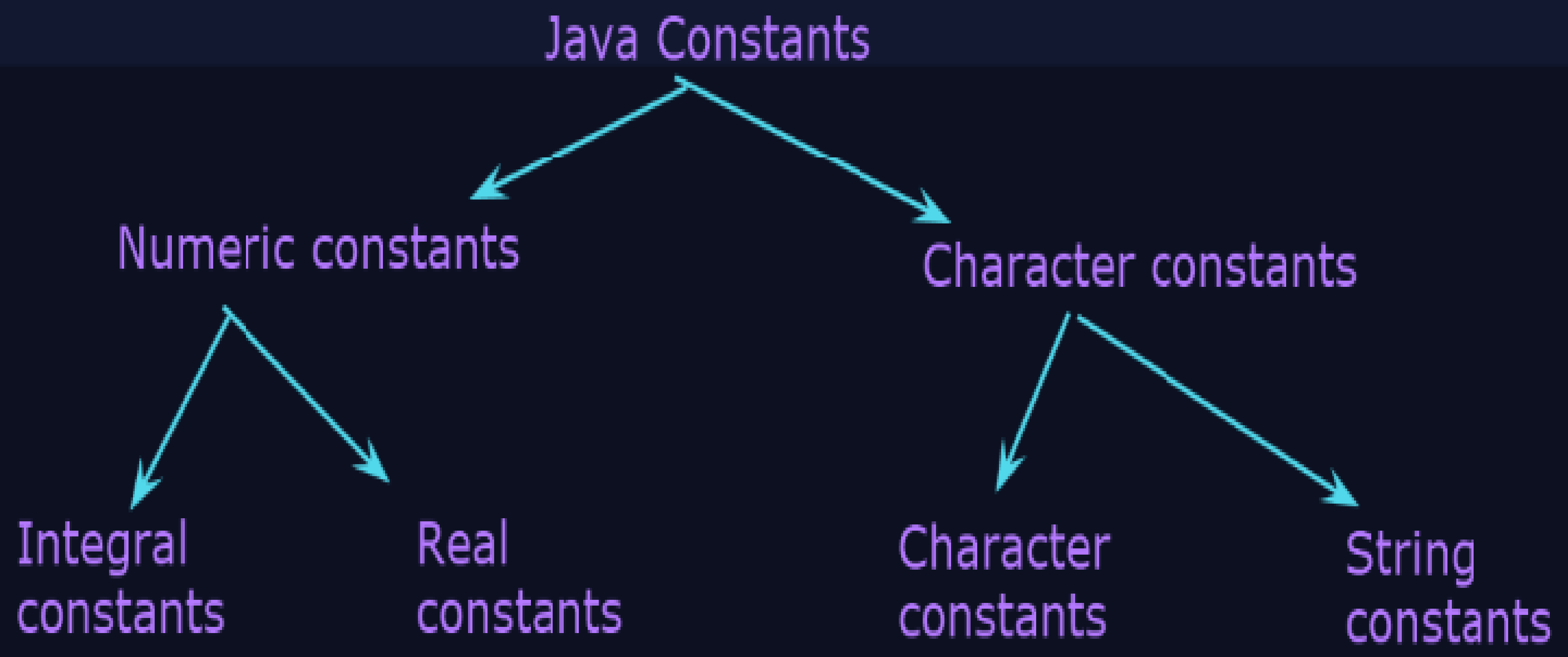
```
4.Operators
5.Separators

Data Types:
```



Quick Summary of Data Types:

Data Type	Size (Bytes)	Range	Default Value	Precision
`byte`	1	-128 to 127	0	-
`short`	2	-32,768 to 32,767	0	-
`int`	4	-2,147,483,648 to 2,147,483,647	0	-
`long`	8	±9.2E18	0L	-
`float`	4	±3.4E38	0.0f	5-6 decimals
`double`	8	±1.7E308	0.0d	14-15 decimals
`char`	2	0 to 65,535 (Unicode)	'\u0000'	-
`boolean`	1 bit	true / false	false	-

Literals: Java Constants

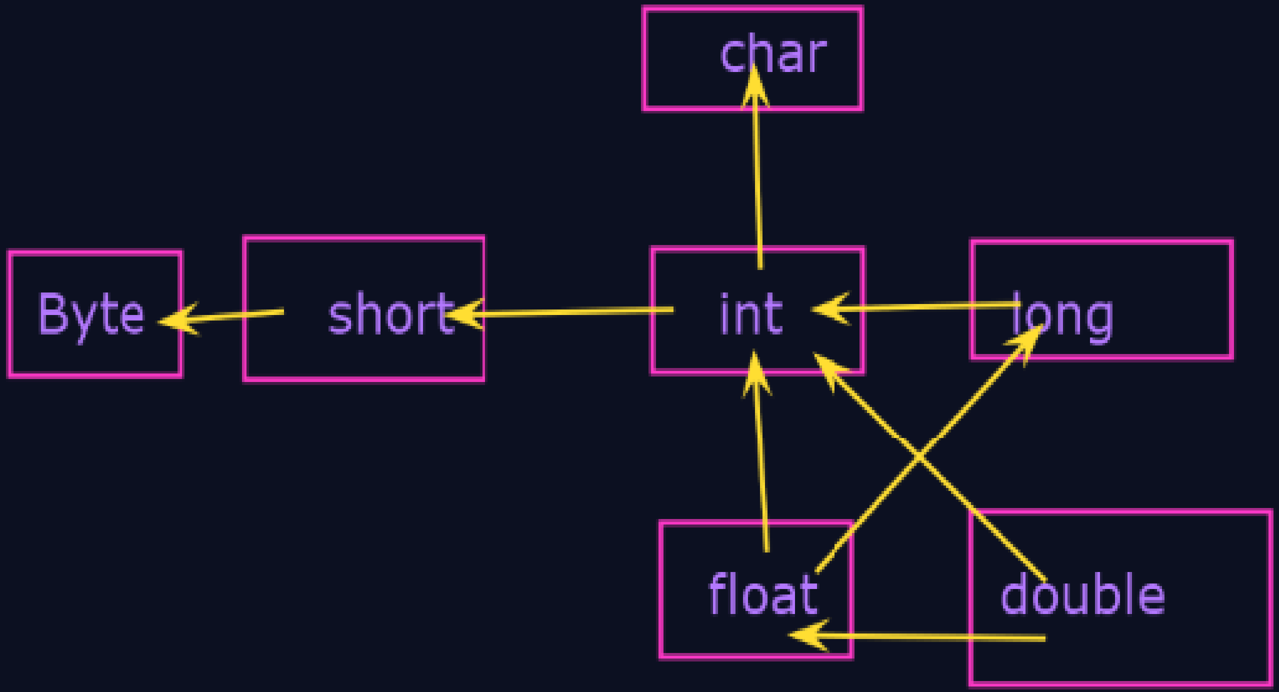


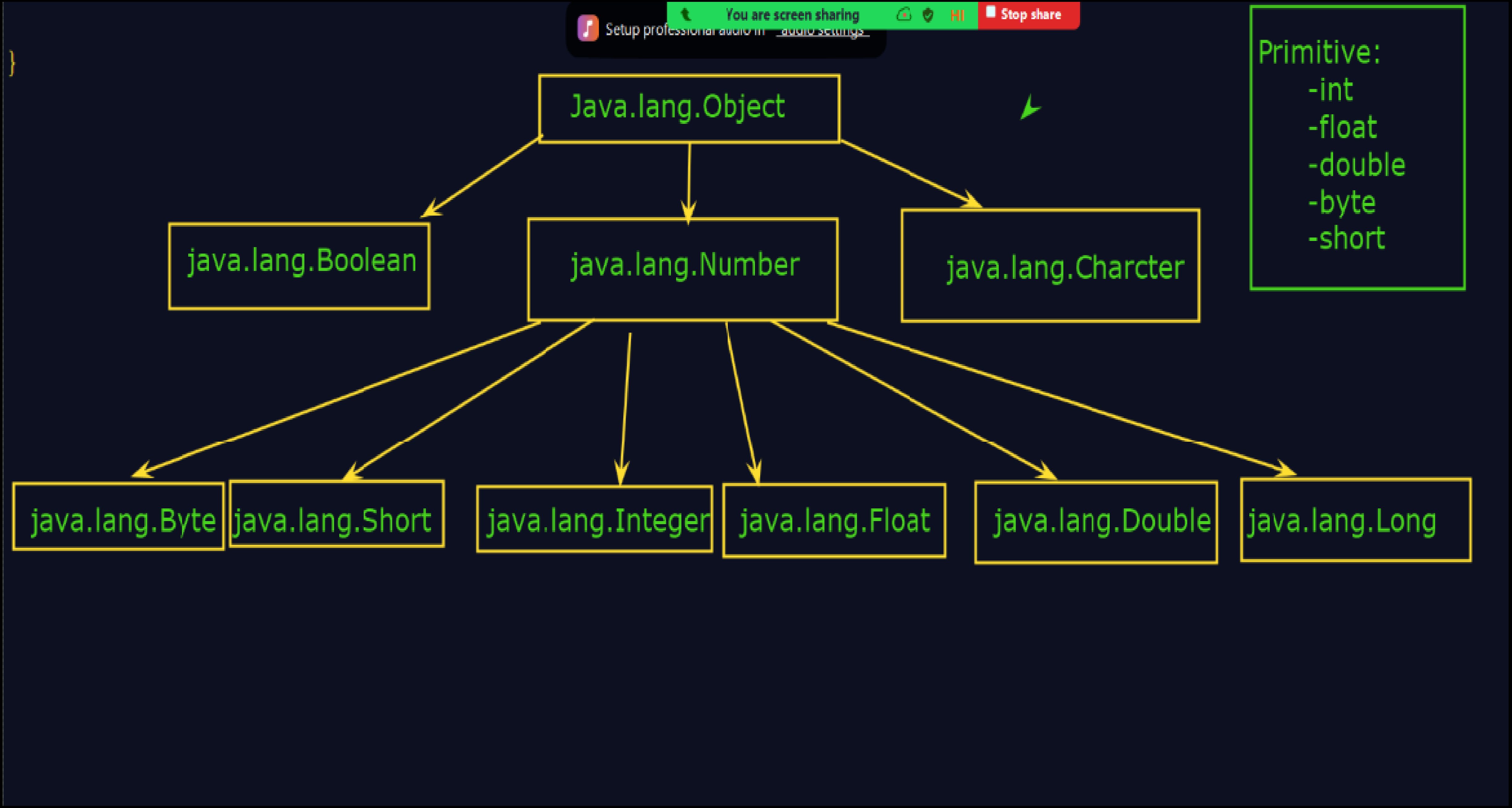
Widening : Upcasting: Implicit casting

-compiler auto manage karta hai

$$\begin{tikzcd}
 & & \text{char} \arrow[d] & \\
\text{Byte} \arrow[r] & \text{short} \arrow[r] & \text{int} \arrow[r] \arrow[d] \arrow[dr] & \text{long} \\
 & & \text{float} \arrow[ur] \arrow[r] & \text{double}
\end{tikzcd}$$

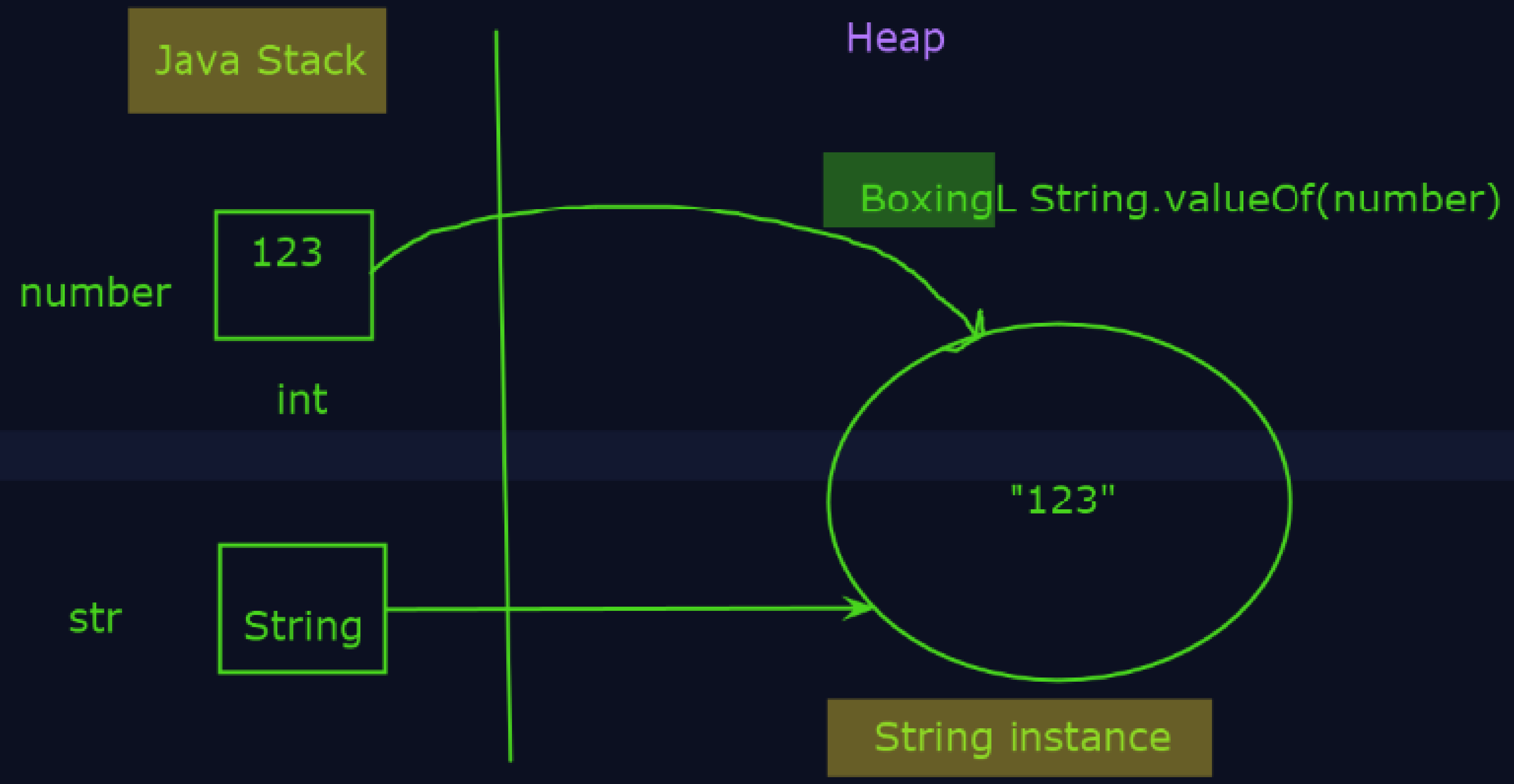
Narrowing : Downcasting : Explicit Casting



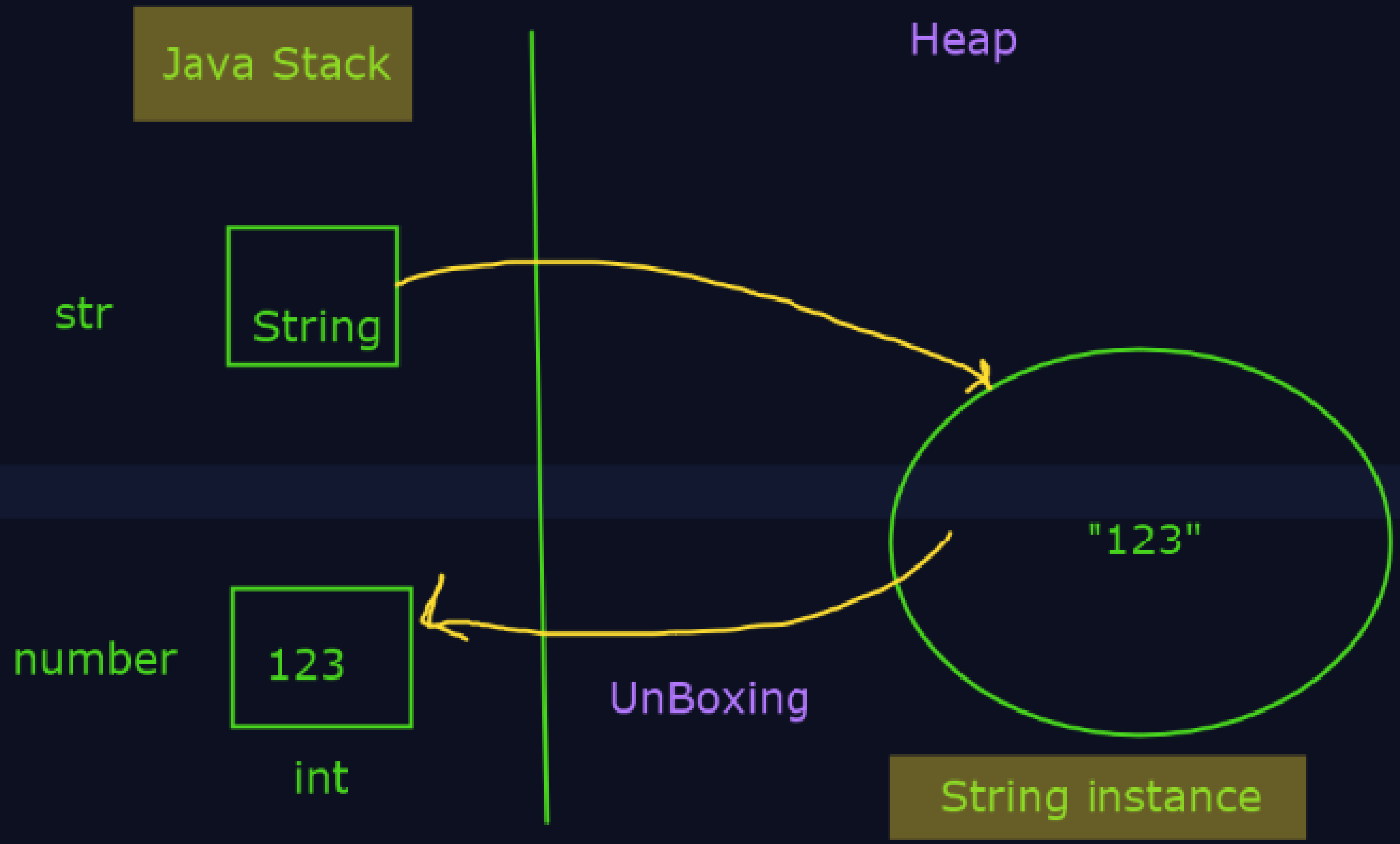


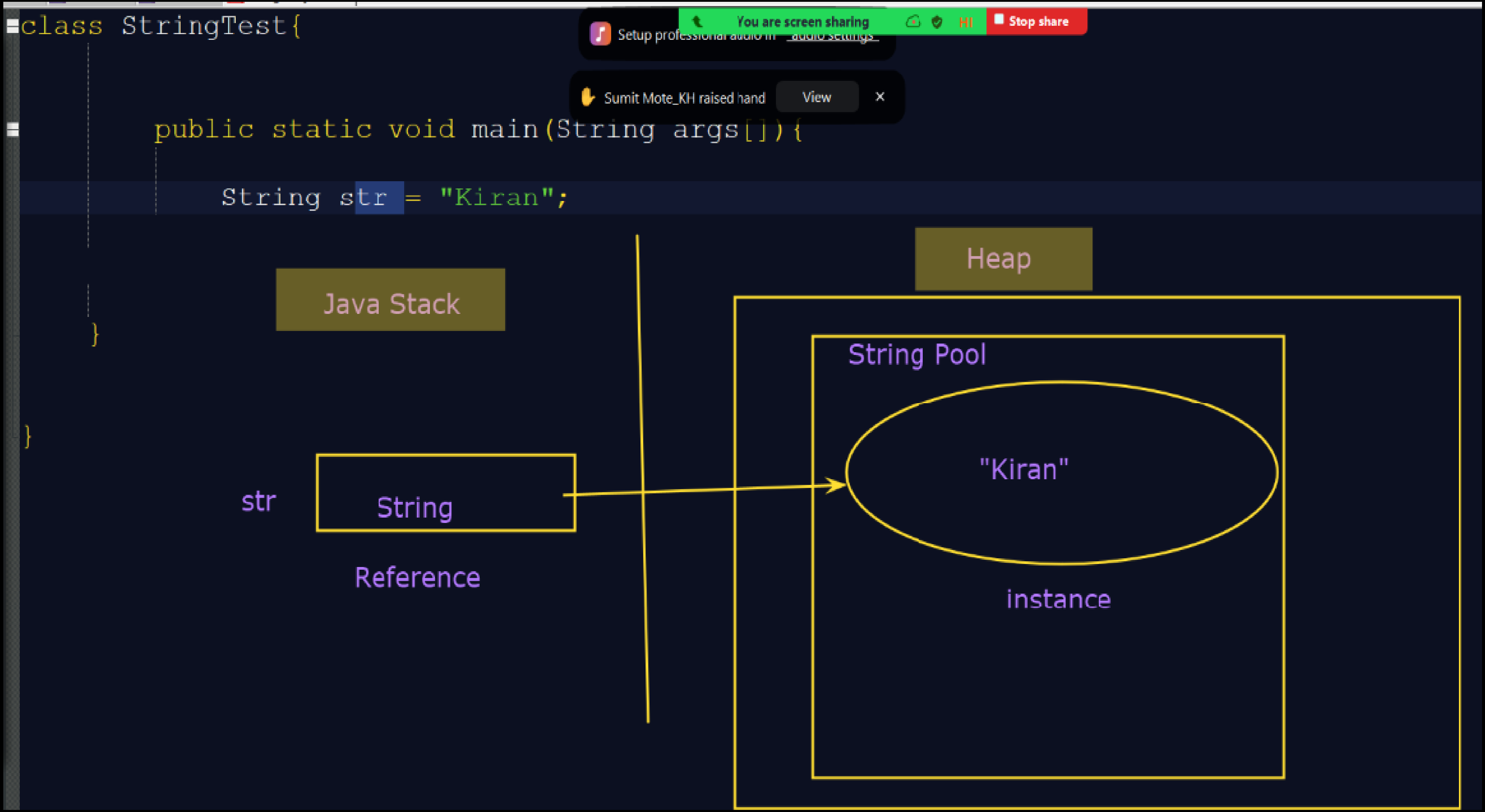
Primitive:
- -int
- -float
- -double
- -byte
- -short

```
int number =123
String str = String.valueOf(number); //Boxing
```



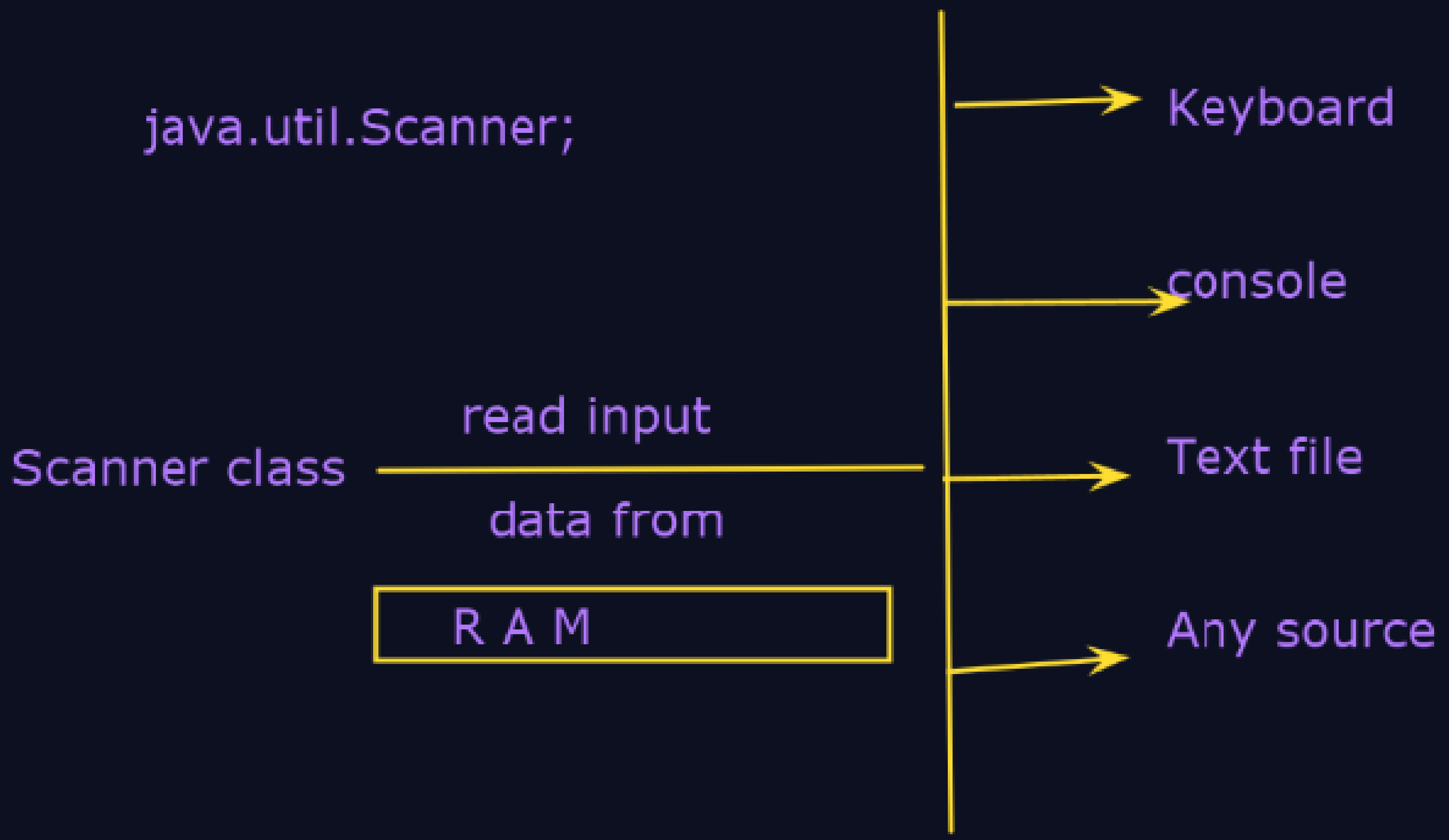
```
String str ="123"
int number = Integer.parseInt(str); //UnBoxing
```

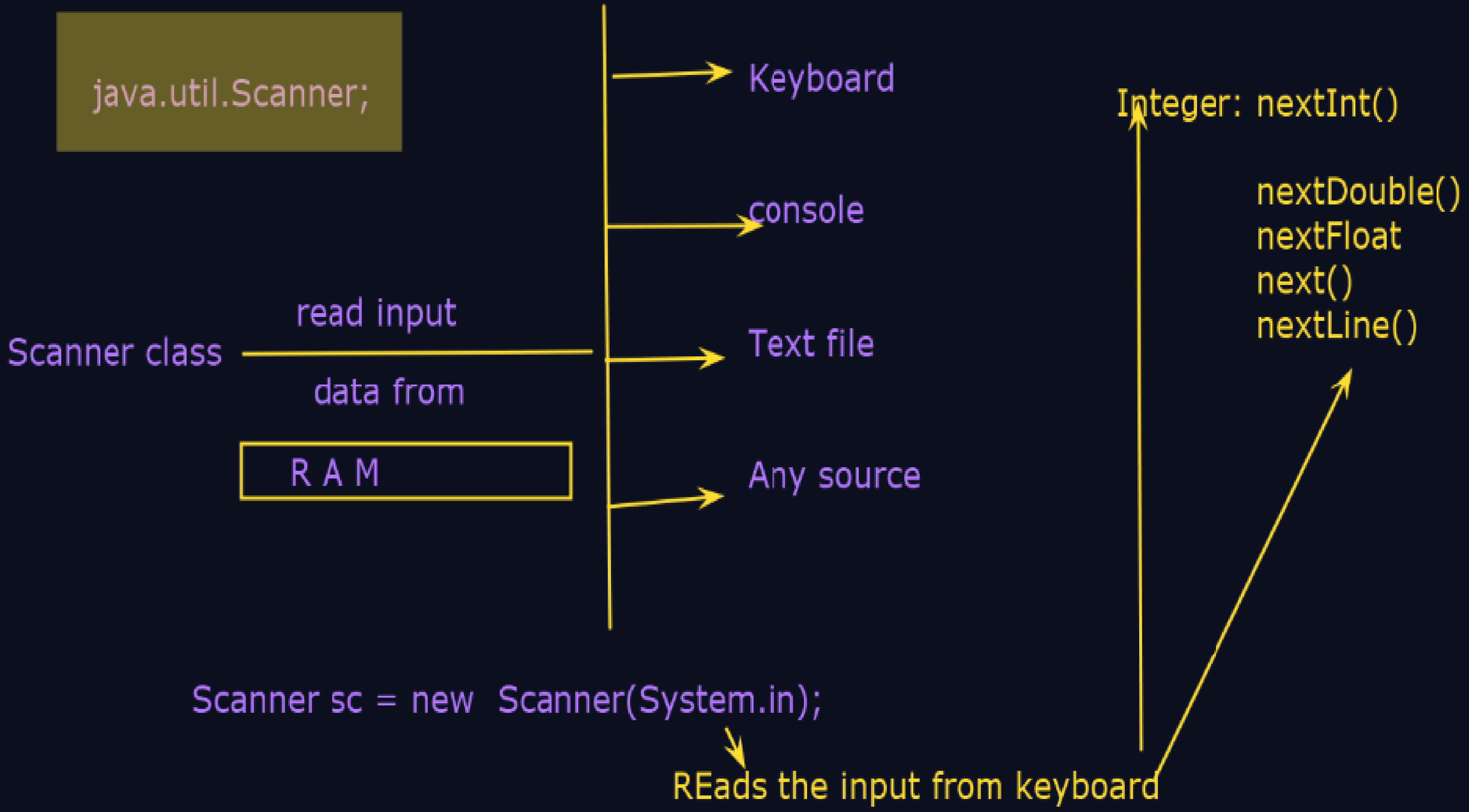




```
class StringTest{


        public static void main(String args[]){

            String str = "Kiran";

        }

}
```

java.util.Scanner;





```
java.util.Scanner;
```

```
Scanner sc = new  Scanner(System.in);
```

Reading Different Types of Input

Reading Different Types of Input

Method	Reads	Example Input
nextInt()	Integer	10
nextDouble()	Double (decimal)	3.14
nextFloat()	Float (decimal)	5.75
nextLong()	Long Integer	123456789
nextBoolean()	Boolean	true / false
next()	Single word	"Hello"
nextLine()	Full line (including spaces)	"Hello World"

Arithmetic Operators

Operator	Description	Example	Output
+	Addition	10 + 5	15
-	Subtraction	10 - 5	5
*	Multiplication	10 * 5	50
/	Division	10 / 5	2
%	Modulus (Remainder)	10 % 3	1

Relational Operators

Operator	Description	Example	Output
==	Equal to	10 == 5	false
!=	Not equal to	10 != 5	true
>	Greater than	10 > 5	true
<	Less than	10 < 5	false
>=	Greater than or equal to	10 >= 5	true
<=	Less than or equal to	10 <= 5	false

Logical Operator

Operator	Description	Example	Output
&&	Logical AND	(10 > 5) && (5 < 10)	true
||	Logical OR	(10 > 5) || (5 < 10)	true
!	Logical NOT	!(10 > 5)	false

Bitwise Operator

Operator	Description	Example	Output
&	Bitwise AND	5 & 3 (0101 & 0011)	1
|	|	Bitwise OR	`5
^	Bitwise XOR	5 ^ 3 (0101 ^ 0011)	6
~	Bitwise NOT	~5 (~0101)	-6
<<	Left Shift	5 << 1	10
>>	Right Shift	5 >> 1	2

Assignment Operator

Operator	Description	Example	Equivalent
=	Assign	x = 5	x = 5
+=	Add and assign	x += 5	x = x + 5
-=	Subtract and assign	x -= 5	x = x - 5
*=	Multiply and assign	x *= 5	x = x * 5
/=	Divide and assign	x /= 5	x = x / 5
%=	Modulus and assign	x %= 5	x = x % 5

Bitwise Operator

Operator	Description	Example	Binary Representation	Output
&	Bitwise AND	5 & 3 (0101 & 0011)	0101 & 0011 = 0001	1
`	`	Bitwise OR	`5	3 (0101
^	Bitwise XOR	5 ^ 3 (0101 ^ 0011)	0101 ^ 0011 = 0110	6
~	Bitwise NOT	~5 (~0101)	~0101 = 1010 (2's complement)	-6
<<	Left Shift	5 << 1	0101 << 1 = 1010	10
>>	Right Shift	5 >> 1	0101 >> 1 = 0010	2

Bitwise Shift Operator

Operator	Operation	Zero Fill?	Used For
x << n	Left Shift	Yes (right-side)	Multiplication by 2^n
x >> n	Right Shift	No (preserves sign bit)	Division by 2^n
>>>	Unsigned Right Shift	Yes (left-side)	Handling unsigned data