Lecture 9
Input & Output in Java

How to take Input from User :-

There are many ways to take input from users but the most commonly used method is Scanner class.

→ We can take input from user using classes like Scanner, BufferedReader or Console.

→ BufferedReader and Console classes methods to take input from user are obselete. Now-a-days most commonly used method is using Scanner class.

NOTE:- If you use console class to take input then it will not work in the ide's like IntelliJ, Eclipse as they don't provide a real console. If you really want to use Console class then you will need to run your Java Program from a terminal

→ Scanner class was introduced in Java 5 (JDK 1.5)

→ Scanner class is located in the java.util package

→ There are so many methods within Scanner class.

e.g:-

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

Constructor

- Scanner → class
- scanner → object name
- new → keyword
- Scanner(System.in) → takes input from keyboard

→
```
int a = scanner.nextInt();
int b = scanner.nextInt();
```

nextInt → scans the next token of the input as an int

~~float x = sc.nextFloat();~~

```
float x = scanner.nextFloat();
```

↓ Used to take float as Input

```
String str = scanner.next();
System.out.println(str);
```

↳ next() method is used to read a single word or token as a String from the input. It only reads until it finds a whitespace, tab or a new line. It reads one word at a time.

↳ In this case if we give input

Jenny khatri

then it will only take

Jenny as input

Imp Points:→ next() reads input until it encounters a delimiter (space, tab, newline)

→ It doesn't read the entire line of input but only the next word (token)

→ It is useful when you want to capture individual tokens from a sentence.

```
String str1 = scanner.nextLine();
```

nextLine() method reads the entire line of input, including spaces until the user presses enter.

```
byte y = scanner.nextByte();
```

↳ scans the next token of the input as a byte. It will throw an InputMismatchException if the next token can not be translated into a valid byte value.

Some other Methods:-

nextShort()

nextLong()

→ One method is :- **hasNextInt()**

↓

returns true if the next token in this scanner's input can be interpreted as an int value. (within int range)

eg:-

```
boolean isInt = scanner.hasNextInt();
System.out.println(isInt);
```

If you input here: → 2147483649 then o/p would be false because the given number is not within the range of int.