Lecture 9

Escape Sequence & Input/output in Java

- **Escape Sequence** :- It is a combination of characters that represents a special character, typically preceded by a backslash (\).

→ Escape sequence are used within Strings to indicate special characters that can not be typed directly.

e.g:- \n → moves the cursor to the next line.
\t → insert a horizontal tab
\\ → inserts a backslash
\" → inserts double quote
\' → inserts a single quote
\b → inserts a backspace
\r → carriage return (moves the cursor to the beginning of the current line without advancing to the next line)
\f → form feed

→ These are also known as backslash character constants.

NOTE:- Each of them represents one character, although they consists of two character

→ These are valid character literals.

→ The Java Compiler interprets these characters as a single character that adds a specific meaning to the compiler

① \n :- new line

e.g:- System.out.println("Hello\n Welcome to Jenny's lectures!");

Output:- Hello
Welcome to Jenny's lectures!

② \t :- inserts a horizontal tab

e.g:- System.out.println("Hello \t Welcome");

Output:- Hello Welcome

③ \" :→ inserts a double quote in the text.

e.g:- System.out.println("Hello, Welcome to \"Jenny's Lectures\".");

Output:- Hello, Welcome to "Jenny's lectures".

④ \' :→ inserts a single quote

we use this escape sequence in cases where directly typing a single quote would cause a syntax error or confusion in the code.

It is necessary when we are working with char literals.

e.g:- char ch = '\'';

while defining a char literal in Java, it must be enclosed in single quotes. If we want to print a single quote character itself then we need to escape it using \', since the plain single quote without backslash (\) would be confused with the end of char literal.

Now we can print this :-

System.out.println(ch);

→ But in Java string literals must be enclosed in double quotes (") So technically we don't need to escape single quote when writing them inside a string.

e.g:- System.out.println("Hello, Welcome to Jenny's lectures!");

[no need to escape this single quote with \.]

But if you want to follow to stylistic consistency, you can use \' here like:- System.out.println("Hello, Welcome to Jenny\'s lectures!");

⑤ \\ :→ inserts a backslash

eg:- System.out.println("\\ - It's a backslash");

Output:- \ - It's a backslash.

⑥ \b :→ backspace [moves the cursor one position back with or without deleting the character (depending on the compiler)]

e.g:- System.out.println("Jenny's Lecturess\b");

Output:- Jenny's Lectures

It is often utilized in console or terminal output to overwrite or erase previously printed characters.

The backspace function (\b) in programming mimics the idea of erasing a character

It is being used to make the console interactive

e.g:- in Progress bar simulation.

in count down timer

(7) \r :- Carriage Return

It moves the cursor to the beginning of the current line, overwriting the output (the text that was previously printed on that line)

e.g:- System.out.println("Hello\r Welcome");

Output :- Welcome

* The name comes from the era of typewriters where the "carriage" would physically return to the starting position of the line

→ In modern applications the \r is commonly used in console or terminal based programs to update the output dynamically on the same line, rather than having multiple lines of output for continuously updated information

→ One of the most common uses of \r is to overwrite a line in the console to simulate a progress bar or countdown.

→ Overwriting temporary output :- (If we want to show the status of a task and overwrite it once the task is complete, we can use \r to update the status dynamically.

e.g:-
```
public class EscapeDemo {
    public static void main(String[] args) throws InterruptedException {
        System.out.print("Live Class would be started at 8:00 PM..");
        Thread.sleep(millis: 2000);
        System.out.print("\r Welcome to Today's Live Class\n");
    }
}
```

(8) \f :- form feed (Not commonly used today, but it was historically used to move to the next page on printers.)

- In modern applications it is rarely used and its behaviour is undefined and in many cases it simply produces no visible effect on modern terminals or consoles.

⑨ \uxxxx :- Unicode character

e.g:- System.out.println("\u0041");

Output:- A