Lecture 7

What we will cover! -

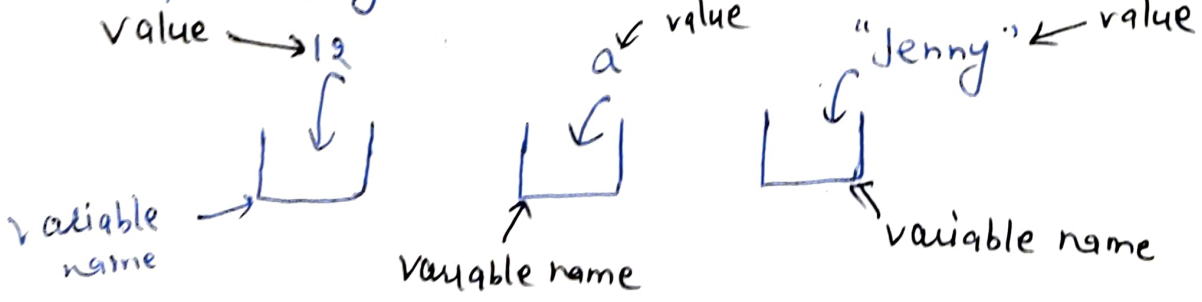
1. what are variables ?
2. What are keywords ?
3. Identifiers in Java
4. Data types
5. Input & Output

example from previous lecture
student class

① **Variables** :- When we write programs or working on projects then we have to deal with data (numbers, characters, etc.) So we need something to store this data in computer memory as computer can't remember anything because computer doesn't has its own brain.
So to store data in computer's memory, variables are required.

→ Basically we can say variables are a way to store information in our computer.

→ In other words we can say variables are like containers used for storing data values.



⇒ Variables are data containers that save the data values during program execution. meaning data is stored in RAM using variables.

e.g:- in our kitchen we have different containers to store different- different things

So in a variable we can store different type of data.

e.g:-
Jenny → a name which is String
123 → a number of type integer
12.31 → a floating-point number
a → a single character
True, false → boolean values

What type of value a variable can hold is decided by its Data Type

So each variable has its own data type (int, float, char, bool etc.). and size of each variables can be different.
[same as ~~in~~ size of kitchen containers]

How to define variables / Defining with initialization

Syntax:- `datatype variable_name = value;` ← initialization of variable

eg:-
```
int a = 10;
float b = 10.10;
String name = "Jenny";
```

<type> <name> [= literal or expression]

literal is raw data (e.g:- 10, 10.10, Jenny)
expression is evaluated to a single value

We can change the value of a variable during our program ~~writing~~ /execution.

e.g:- a = 15; | Now the value of a becomes 15.

At first a value was 10, after that we change the value to 15. So Now in a we have 15.

→ It is called a variable because the value inside it can change. But redeclaration of variable (within same scope) is not allowed in Java

→ We can access the variable by its name

Declaration of a variable / Defining a variable without initialization.

Syntax:- <Type><name>;

e.g:- int num;

We can declare more than one variable in single line.

<Type><name1, name2 ------->

e.g:- int num1, num2;

Assigning a value to a variable :- <name> = literal or expression

num1 = 10;

after declaration we can assign value using an assignment operator

NOTE:- We have to declare a variable before its use.

NOTE:- Declaration means only telling the compiler about the type & name of variable, no space/memory would be allocated to it. But this happen only when we use external variables like

extern int a; (Declaration only, no memory allocation)

& this concept is in C/C++, not in Java

→ it tells the compiler that x exists somewhere else, likely in another file or further down in the same file.

Definition means asking compiler to allocate space for variable.

int x; (4 byte space allocated in memory)

⇒ The amount of space allocated depends on the datatype of that variable. for int its 4 bytes on many systems.

Variables are also of different types!-

- Local variable
- Instance variable / class-level variables
- Static variable

local variable :- within a block or within a method.

eg:-
```
class VariableDemo
{
    public static void main(String[] args).
    {
        int x;        → local variable Definition
        x = 10;       → assigning value to x
        System.out.print(x);
    }
}
```

Declaration + definition

x is local variable to function main. within main we can use this variable.

Instance Variable :-

eg:-
```
class VariableDemo
{
    int x;   → no memory allocated at this time.
    ... } more code
}
```

here memory would be allocated to x when an object of this class is created.

Static variables :-
```
class VariableDemo
{
    static int x;  - Declaration + Definition
                     [memory allocated to x & initialized to 0]
    ...
}
```

Here x is static variable, meaning memory is allocated when the class VariableDemo is loaded, not when an object is created.

NOTE:- In Java there is no "declaration only". Declaration always allocates memory so its always a definition.

Definition:-	int x;
Assignment:-	x = 10;
Initialization:-	int x = 10;

every variable has a datatype associated with it e.g:- int, float etc

NOTE → Local variables in Java are not automatically assigned default values. Compiler never assigns a default value to an uninitialized local variable. If you can't initialize your local variable where it is declared, make sure to assign it a value before you attempt to use it.

Accessing an uninitialized local variable will result in a compile-time error.

NOTE → But in Java, instance & static variables are implicitly initialized with default values

Naming Rule for Variable:-

* Names can contain letters, digits, underscore (_) & dollar sign
  e.g. a, num,
* Names must begin with a letter. It can also begin with underscore (_) or $ (dollar) but it's not recommanded.
* Names are case sensitive (e.g. rollno, Rollno are different)
* should follow Camelcase (lowerCamelCase) convention. Should start with lower case & each subsequent word starts with Capital letter (eg:- myVariable)

* Names should start with a lowercase letter and it can't contain any whitespace.
* Reserved words can not be used as names.
* First letter can't be a digit
* There is no limit on length of a variable but variable name should be short yet meaningful. e.g:- age, lastName, isCorrect.

Valid variable names:- a, num, sum, _num, $sum, roll_no, id123, Roll$, Int, Else

Invalid variable names:- 1sum, 123, void, abstract, roll no, int, -(underscore), else

Keywords:- Keywords are predefined words in a programming language that have a special meaning.

- keywords can not be used as identifiers (like variable name, class name, method names etc.). e.g:- in real-life, red light → stop
  push, pull etc.
  keywords are reserved by the language for specific functionalities
  e.g:- class, if, int, else, abstract, for, public etc.
- Reserved words include both keywords and other words that the language has reserved for future use or special words that are not yet functional in the language
  e.g:- in Java we have some reserved words that are not keywords but are reserved for future use.
  goto & const (not used but reserved for future)

Reserved words can not be used as identifiers.

> Reserved words = keywords + other words reserved for future use.

Data Type :-

Identifiers :- An identifier is a name given to various elements in the program such as variables, methods, classes, packages and interfaces.

Identifiers are symbolic names used for identification. They can be variable name, method name, class name.

Rules :-

- Identifiers can be composed of letters (a-z), (A-Z), digits (0-9), underscore (_) and dollar sign ($).
- Cannot start with a digit
- Must begin with a letter, underscore or dollar
- names are case sensitive (e.g. myName & MyName are two different identifiers)
- keywords & reserved words cannot be used as identifier names

Best Practices for naming identifiers :-

- Use short yet meaningful name

  e.g:- `int studentAge;` → name / identifier / variable name
  `float studentMarks;` (Data type)

- Use camel case for variables & methods names.
- Use Pascal case (Upper Camel Case) for class names

Practice Time :- check following statements are valid or not:-

`int a = 1+2;`	valid	because we can write expressions as well as literals on R.H.S. Expressions would be evaluated to a single value & that value would be assigned to L.H.S (to the variable)
`int b = a+3;`	valid	
`int c = b/2;`	valid	
`int d = a+b;`	valid	

Literals :-

- Literals in Java are fixed values that are directly represented in source code.

Literals are raw data.

→ These values do not change during program execution and are directly assigned to variables.

Types of literals :-

① Integer literals :- represent whole numbers

eg:- int age = 15;

↳ can be written in different number systems :-

→ decimal (base 10) : int num = 50; (in decimal its 31)

→ Octal (base 8) :- int num = 037; (starts with 0)

→ Hexadecimal (base 16) :- int num = 0x23, 0x9f, 0xabc

(in decimal 35)

② Floating point literals :- numbers with a decimal point

eg:- double price = 99.99;

NOTE :- By default floating point literals are double. To define a float literal you must append f or F

eg:- float price = 99.9f

③ character literals :- represents single character enclosed in single quotes

eg:- char grade = 'A';

There are some backslash character literals as well like :- '\n' (new line)

'\t' ⇒ tab

'\\' ⇒ backslash etc.

~~Runtime Variable Declaration~~

④ String literals:- represents sequence of characters enclosed in double quotes.

e.g:- String name = "Jenny";

⑤ Boolean literals:- represents truth values true or false

e.g:- boolean isCorrect = true;

⑥ Null literal:- represents absence of an object reference.

eg:- String name = null; [String is reference data type in Java]

[As null type has no name, it is impossible to declare a ~~value~~ variable of the null type or to cast to the null type]

Expressions in Java:-

An expression is a combination of values, variables, operators & method invocations that are evaluated to produce a single value. An expression can be a single value or a combination of values that produce a result.

```
eg:- int x = (1+2) + (3*5);
     int x = x+1;

boolean isCorrect = true;
boolean isJavaEasy = false;
boolean result = (isCorrect && isJavaEasy);
```

Constants in Java :-

Constants are variables whose values can not be changed once they are assigned.

→ declared using 'final' keyword which prevents modifications to their value.

→ Once a Constant is initialized, its value is set for the lifetime of the program.

e.g:-

```
final int MAX_VALUE = 100;
final double PI = 3.14159;
```

→ Constants are valuable for storing fixed values like configuration settings, mathematical constants (PI) or commonly used values.

→ Conventionally, constants are named in uppercase with underscores separating words like MAX_VALUE
DAYS_IN_WEEK

Why Constants:-

- It makes the code more readable as they clearly define fixed values that are easily understood

Maintainability:- If a constant needs to be changed, it can be done in one place without the need to change it in multiple locations in the code.

IMP Points:- Constants must be initialized when declared. & once initialized, they can not be changed.

Some More info on Variables :-

Variables can store 2 type of things :-

- Primitive type
- references

Primitive hold fundamental values including integers, booleans, floating point numbers.

Object references hold references to objects.

e.g.:-

```
int x = 10;
float marks = 98.99f;
double average = 67.678;
char choice = 'a';
boolean flag = true;
boolean powerOn;
powerOn = flag;
long big = 1256896L;
int y = x;
```

```
class StudentInfo {
    . . .
}

StudentInfo s = new StudentInfo();
```

↓

object reference

Practice Time :-

```
public class Student
{
    int rollNo = 101;
    String name = "Jenny";
    float marks = 98.99f;
    void display()
    {
        System.out.println("The Student " + name + "has" +
            "marks");
    }
    public static void main(String[] args)
    {
        Student s = new Student();
        s.display();
    }
}
```

```
int rollNo;
rollNo = 101;
```

} ⇒ will give error. assignments not allowed at class level variables

this + symbol acts as concatenation here. It will append the right side String/value to the left side String/value