Language Fundamentals

1. **Identifiers**
2. **Literals**
3. **Keywords/Reserved words**
4. **Operators**

1. Identifiers

Identifiers are symbolic names used to uniquely identify programming elements such as variables, functions, classes, and other entities in a program. They play a crucial role in making the code more readable, maintainable, and understandable by providing meaningful names to different elements, aiding both programmers and the compiler in effectively communicating the program's logic and structure.

OR

Identifier is a name assigned to the programming elements like variables, methods, classes, abstract classes, interfaces.

Genie Ashwani

Rules and regulations For Identifiers:

- Identifiers should not be started with any number
- identifiers may be started with an alphabet, '_' symbol, '$' symbol,

Examples:

Valid Identifiers

1. userName
2. calculateSum
3. _counter
4. $totalAmount
5. StudentInfo
6. MAX_VALUE
7. isPrime
8. userInput
9. PI
10. Book

Invalid Identifiers

11. 123numbers (starts with a digit)
12. my-variable (contains a hyphen)
13. illegal@char (contains special characters)
14. class (a reserved keyword)
15. 3DArray (starts with a digit)
16. my space (contains whitespace)
17. void (a reserved keyword)
18. if (a reserved keyword)
19. @symbol (starts with a special character)
20. return (a reserved keyword)

Genie Ashwani

Questions(Home Work):

```
int salary=10000
int 11salary=999
String _addr="Agra"
float $sal=10000.0f
String student11No="V132-1111"
String std_Name="Ashwani Upadhyay"
float emp$Sal=50000.0f
```

NOTE: Identifiers are not allowing all operators and all special symbols except '_' and '$' symbols

More Examples:

```
int stdNo=111; ========= valid
int std+No=111;=========Invalid
String std*Name="Ashwani";===== Invalid
String #stdAddr="Delhi";====Invalid
String std@id="123";===Invalid
float std.Fee=50000.0f;=====Invalid
String std-Addr="Hyd";======Invalid
String std_Addr="Hyd";======Valid
```

Note 2: Identifiers are not allowing spaces in the middle.

Valid Identifiers:

- myVariable
- totalAmount
- userInput
- className
- isPrime

Genie Ashwani

Invalid Identifiers:

- my Variable (contains a space)
- illegal char (contains a space)
- user input (contains a space)
- class name (contains a space)
- invalid identifier (contains a space)

Note 3: Identifiers should not be duplicated with in the same scope, identifiers may be duplicated in two different scopes.

```
class Student {
        int num=10;-----> Class level
        short num=20;----> Error
        double sal=33.33---> No Error
        void print() {
                float sal=22.22f; -----> local variable
                double fsal=33.33;---> Error
                long num=30;---> No Error
        }
}
```

Note 4: In java applications, we can use all predefined class names and interface names as identifiers.

Genie Ashwani

Examples:

1. String
2. List
3. Scanner
4. Math
5. Runnable
6. Exception
7. String

2. Literals

literals are fixed values that are directly used in your code. They are used to represent specific data types like numbers, characters, strings, and Boolean values. Literals make your code more readable and help you initialize variables and constants with specific values. Let's explore the different types of literals in Java.

OR

Literal is a constant assigned to the variables

Example
int a=10;
int ----> data types
a ------> variables/ identifier
= ------> Operator
10 -----> constant [Literal].
; ------> Special symbol.

Genie Ashwani

Numeric Literals

Numeric literals represent numerical values. They can be categorized into integers and floating-point numbers.

Integers

An integer literal is a whole number without a decimal point. It can be written in different number bases

Decimal: Default base. Example: 123

Binary: Prefixed with 0b or 0B. Example: 0b1010 represents 10.

Octal: Prefixed with 0. Example: 075 represents 61.

Hexadecimal: Prefixed with 0x or 0X. Example: 0x1A represents 26.

Genie Ashwani

Examples:

```
int decimal = 123;

int binary = 0b1010;

int octal = 075;

int hexadecimal = 0x1A;
```

Floating-Point Numbers

A floating-point literal represents a real number with a decimal point. It can be written in standard or scientific notation.

Examples:

```
double normal = 3.14;

float small = 2.0f;
```

Character and String Literals

Character literals represent single characters enclosed in single quotes ' '.

Examples:

```
char letterA = 'A';

char digit5 = '5';
```

String Literals

Genie Ashwani

String literals represent sequences of characters enclosed in double quotes " ".

Examples:

String message = "Hello, World!";

String empty = "";

Boolean Literals

Boolean literals represent the two truth values: true and false.

Examples:

boolean isRaining = true;

boolean hasCoffee = false;

Genie Ashwani

Escape Sequences

Escape sequences are used to represent special characters within character and string literals. They are written as a backslash \ followed by a character.

Common escape sequences include:

\' for single quote

\" for double quote

\\ for backslash

\n for newline

\t for tab

\r for carriage return

\b for backspace

Examples:

```
char singleQuote = '\'';
String quoteExample = "She said, \"Hello!\"";
String newLine = "First line.\nSecond line.";
```

Null Literal

The null literal represents the absence of a value. It is often used to indicate that a reference does not refer to any object.

Genie Ashwani

Example:

```
String name = null;
```

Genie Ashwani