* **Boilerplate code**

|  |
| --- |
| public class JavaBasics {  public static void main(String args []) {  System.out.print("Hello Sheetal!");  }  } |

* **Literals** are those values that doesn’t changes

i.e., any number: 1,2,3, 4... or 'a', 'b', 'c'... or '@','\*'

* **Variables** are those whose values varies i.e., int a=10; etc.
* Names of variables & classes and functions etc. are **Identifiers**.

i.e., a, b, class, main, println…

Data types:

|  |  |
| --- | --- |
| **Primitive (**Already Exist) | **Non-Primitive (**User Created) |
| * + - Byte     - Short     - Char     - Boolean     - Int     - Long     - Float     - Double | * + - String     - Array     - Class     - Object     - Interface |

* Java, C++, Python are **typed** languages (Where one has to define the type of data input), Whereas JS is not.
* Size of **Data types:**

|  |  |
| --- | --- |
| **Data types** | **Size** |
| * Byte * Short * Char * Boolean * Int * Long * Float * Double | * 1 byte [-128 to 127] * 2 bytes * 2 bytes [‘a’ to ‘z’, ‘A’-’Z’, ‘@’...] * 1 byte [True or False] * 4 bytes [-2B-2B] * 8 bytes * 4 bytes * 8 bytes |

* **Comments** in Java is added by typing ‘//’ or ‘/\*…. \*/’ (Multiline comment).
* **Input** in Java

|  |
| --- |
| * next * nextLine * nextInt * nextByte * nextFloat * nextDouble * nextBoolean * nextShort * nextLong |

* **Type Conversion**  
  also knows as widening conversion & Implicit conversion.  
  Conversion happens when:  
  a) Type compatible  
  b) Destination type > source type  
    
  float number = sy.nextInt();

byte>>short>>int>>float>>long>>double

conversion from int -> float isn't possible

whereas from float-> int is possible

* **Type Casting**  
  (Also called narrowing conversion & explicit conversion) is forcefully conversion irrespective of lossy conversion.
* **Type promotion**  
  a) java automatically promotes each byte, short, or char operand to int when evaluating the expression  
  b) if one operand is long, float or double the whole expression is promoted to long, float or double respectively

Operators

* Symbols that tell compilers to perform some operation.  
  i.e., sum = a+b;  
  here a & b are operands, whereas '+' is an operator.
* **Types of operators**
  + Arithmetic operator
  + Relational operator
  + Logical operator
  + Bitwise operator
  + Assignment operator
* Arithmetic operator:

|  |  |  |
| --- | --- | --- |
| Binary (Have two operands) | | Unary (Only single operand) |
| + - \* / % | (A+B) (A-B) (A\*B) (A/B) (A%B) | ++ (++A, A++) -- (--A, A--) |

* + Unary Operators
    - Increment operators (++)
      * Pre-increment(++A): value changes first then we use the value
      * Post-increment(A++): value use first then we change the value
    - Decrement operators (--)
      * Pre- Decrement (--A): value changes first then we use the value
      * Post-Decrement (A--): value use first then we change the value
* Relational operator :

|  |  |
| --- | --- |
| ==  ! =  >  <  >=  <= | to check whether values are equal  to check whether values are not equal  checks greater than  checks less than  Greater than equal to  Less than equal to |

* Logical Operator
  + &&(Logical AND): Based on AND Logic Gate condition; True Only when all statement are true.
  + || (Logical OR): Based on OR Logic Gate condition; false Only when all statement are false.
  + ! (Logical NOT): Based on NOT Logic Gate condition; makes false condition true and   
    true to false. Only one statement is required (minimum)